**Advanced Database Systems**

**Sample Program list**

Q.1 Write Java-MySQL program for horizontal fragmentation (client–server).

Q.2 Write Java-MySQL program for vertical fragmentation (client–server).

Q.3 Write Java-MySQL program for Semijoin.

**Q.4 a) Write PLSQL Procedure to find given number is prime or not.**

**create or replace procedure primepro(a in number)**

**is**

**i number;**

**flag number;**

**begin**

**i:=2;**

**flag:=1;**

**for i in 2..a/2**

**loop**

**if(mod(a,i)=0)**

**then**

**flag:=0;**

**exit;**

**end if;**

**end loop;**

**if(flag=1)**

**then**

**dbms\_output.put\_line("prime")**

**else**

**dbms\_output.put\_line("not prime")**

**end if;**

**end primepro;**

**declare**

**a number:=&a;**

**begin**

**primepro(a);**

**end;**

**b) Write PLSQL Procedure to find factorial of given number.**

**create or replace procedure fact(num in number)**

**fac number:=1;**

**n number:=num;**

**begin**

**while n>0**

**loop**

**fac:=n\*fac;**

**n:=n-1;**

**end loop**

**dbms\_output.put\_line(fac);**

**end ;**

**Q.5 a) Write PLSQL Procedure to find reverse of given number.**

**create or replace procedure revnum(num1 in number)**

**is**

**result number:=0;**

**copy number:=num1;**

**begin**

**while(copy>0)**

**loop**

**result:=result\*10+mod(copy,10);**

**copy:=floor(copy/10);**

**end loop;**

**dbms\_output.put\_line(result);**

**end;**

**b) Write PLSQL function to find factorial of given number.**

**create or replace function fact(num in number)**

**return number**

**is**

**fac number:=1;**

**n number:=num;**

**begin**

**while n>0**

**loop**

**fac:=n\*fac;**

**n:=n-1;**

**end loop**

**return fac;**

**end fact;**

**Q.6 a) Write PLSQL function to find reverse of given String.**

**create or replace function revstr(str in string)**

**return string**

**is**

**str1 varchar2(20);**

**len number;**

**ch varchar2(1);**

**i number;**

**begin**

**len:=length(str);**

**for i in reverse 1..len**

**loop**

**ch:=substr(str,i,1);**

**str1:=str1||ch;**

**end loop**

**return str1;**

**end revstr;**

**/**

**declare**

**str varchar2(40):='&str';**

**res varchar(40);**

**begin**

**res:=revstr(str);**

**dbms\_output.put\_line(res)**

**end;**

**b) Write PLSQL procedure to find average of given numbers.**

**create or replace procedure avg3(a in number,b in number,c in number)**

**is**

**sumof3 number;**

**avgval number;**

**begin**

**sumof3:=a+b+c;**

**avgval:=sum0f3/3;**

**dbms\_output.put\_line(avgval);**

**end;**

**Q. 7 a) Write PLSQL procedure to find greater from three numbers.**

**create or replace procedure greaterpro3(a in number, b in number, c in number)**

**is**

**begin**

**if((a>b) and (a>c))**

**then**

**dbms\_output.put\_line(a ||' is greater');**

**else if((b>a )and (b>c))**

**then**

**dbms\_output.put\_line(b ||' is greater');;**

**else**

**dbms\_output.put\_line(c ||' is greater');**

**end if;**

**end if;**

**end greaterpro3;**

**/**

**//Procedure call**

**declare**

**a number:=&a;**

**b number:=&b;**

**c number:=&c;**

**begin**

**greaterpro3(a,b,c);**

**end;**

**b) Write a cursor that will increase salary by 10% if salary is > 10000 otherwise by 5%.**

**declare**

**empid emp.eid %type;**

**esal emp.esalary %type;**

**cursor s1 is select eid,esalary from emp;**

**begin**

**open s1;**

**loop**

**fetch s1 into empid,esal;**

**if(esal>10000)**

**then**

**esal:=esal\*1.10;**

**else**

**esal:=esal\*1.06;**

**end if;**

**update emp set esalary=esal where eid=empid;**

**exit when s1%notfound;**

**end loop;**

**close s1;**

**end;**

**Q.8 Solve Following Queries on MongoDB**

**Consider the following**

**Employee (empid, name, salary, designation)**

**Create employee collection.**

1. Insert at least 5 records with meaningful data.
2. Display all employees. **db.Employee.find().pretty()**
3. Find the employees whose salary is >10000. **db.Employee.find({salary:{$gt:10000}}).pretty();**
4. Display the employee name whose having salary either 15000 or 25000 **db.Employee.find({$or:[{salary:15000},{salary:25000}]}.pretty();**
5. Update designation of employee 11 to ‘Asst. Prof.’ **db.Employee.update({empid:11},{$set:{designation:"Asst.Prof"}});**
6. Delete the employees having designation “supervisor”. **db.Employee.remove({designation:"supervisor"});**
7. Display top 2 employees having highest salary. **db.Employee.find().sort({salary:-1}).limit(2);**
8. Increment all employee salary by 1000. **db.Employee.update({},{$inc:{salary:1000}},{multi:true});**
9. Increment all employee salary by 6%. **db.Employee.update({},{$mul:{salary:1.06}},{multi:true})**

**Answers:**

**MongoDB Installation wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc | sudo aptkey add - echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/4.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.4.list deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/4.4 multiverse sudo apt-get update sudo apt-get install -y mongodb mongo > db test > use sql; switched to db sql > db.createCollection("Emp") { "ok" : 1 } > db.Emp.insert({EmpId: 1, Name: 'Sweta', salary: 20000, dept\_name: 'sales'}); WriteResult({ "nInserted" : 1 }) > db.Emp.insert({EmpId: 2, Name: 'Aditi', salary: 25000, dept\_name: 'textile'}); WriteResult({ "nInserted" : 1 }) > db.Emp.insert({EmpId: 3, Name: 'dummy2', salary: 30000, dept\_name: 'computer science'}); WriteResult({ "nInserted" : 1 }) > db.Emp.insert({EmpId: 4, Name: 'dummy3', salary: 35000, dept\_name: 'science'}); WriteResult({ "nInserted" : 1 }) > db.Emp.insert({EmpId: 5, Name: 'dummy4', salary: 40000, dept\_name: 'electronics'}); WriteResult({ "nInserted" : 1 }) > db.Emp.find(); { "\_id" : ObjectId("637cb5fe39581d9a77f745ea"), "EmpId" : 1, "Name" : "Sweta", "salary" : 20000, "dept\_name" : "sales" } { "\_id" : ObjectId("637cb67c39581d9a77f745eb"), "EmpId" : 2, "Name" : "Aditi", "salary" : 25000, "dept\_name" : "textile" } { "\_id" : ObjectId("637cb69f39581d9a77f745ec"), "EmpId" : 3, "Name" : "dummy2", "salary" : 30000, "dept\_name" : "computer science" } { "\_id" : ObjectId("637cb6b739581d9a77f745ed"), "EmpId" : 4, "Name" : "dummy3", "salary" : 35000, "dept\_name" : "science" } { "\_id" : ObjectId("637cb6d839581d9a77f745ee"), "EmpId" : 5, "Name" : "dummy4", "salary" : 40000, "dept\_name" : "electronics" } > use sql; switched to db sql > db.Emp.find() { "\_id" : ObjectId("637cb5fe39581d9a77f745ea"), "EmpId" : 1, "Name" : "Sweta", "salary" : 20000, "dept\_name" : "sales" } { "\_id" : ObjectId("637cb67c39581d9a77f745eb"), "EmpId" : 2, "Name" : "Aditi", "salary" : 25000, "dept\_name" : "textile" } { "\_id" : ObjectId("637cb69f39581d9a77f745ec"), "EmpId" : 3, "Name" : "dummy2", "salary" : 30000, "dept\_name" : "computer science" } { "\_id" : ObjectId("637cb6b739581d9a77f745ed"), "EmpId" : 4, "Name" : "dummy3", "salary" : 35000, "dept\_name" : "science" } { "\_id" : ObjectId("637cb6d839581d9a77f745ee"), "EmpId" : 5, "Name" : "dummy4", "salary" : 40000, "dept\_name" : "electronics" } > use Student1; switched to db Student1 > db Student1 > show dbs admin (empty) local 0.078GB sql 0.078GB test (empty) > db.createCollection("Student") { "ok" : 1 } > show collections Student system.indexes > db.Student.insert({Rollno:1, Name:'Ram', Marks:90}); WriteResult({ "nInserted" : 1 }) > db.Student.find(); { "\_id" : ObjectId("637cb8fd7ba759d6b9e3c0ca"), "Rollno" : 1, "Name" : "Ram", "Marks" : 90 } > db.Student.insert({Rollno:2, Name: 'Sachin', Marks:78}); WriteResult({ "nInserted" : 1 }) > db.Student.find(); { "\_id" : ObjectId("637cb8fd7ba759d6b9e3c0ca"), "Rollno" : 1, "Name" : "Ram", "Marks" : 90 } { "\_id" : ObjectId("637cb93a7ba759d6b9e3c0cb"), "Rollno" : 2, "Name" : "Sachin", "Marks" : 78 } > db.Student.findOne({Rollno:1}) { "\_id" : ObjectId("637cb8fd7ba759d6b9e3c0ca"), "Rollno" : 1, "Name" : "Ram", "Marks" : 90 } > show collections Student system.indexes > db.Student.find().pretty(); { "\_id" : ObjectId("637cb8fd7ba759d6b9e3c0ca"), "Rollno" : 1, "Name" : "Ram", "Marks" : 90 } { "\_id" : ObjectId("637cb93a7ba759d6b9e3c0cb"), "Rollno" : 2, "Name" : "Sachin", "Marks" : 78 } > db.Student.find({Rollno:1}).pretty(); { "\_id" : ObjectId("637cb8fd7ba759d6b9e3c0ca"), "Rollno" : 1, "Name" : "Ram", "Marks" : 90 } > db.Student.find({$or:[{Rollno:1},{Rollno:2}]}).pretty() { "\_id" : ObjectId("637cb8fd7ba759d6b9e3c0ca"), "Rollno" : 1, "Name" : "Ram", "Marks" : 90 } { "\_id" : ObjectId("637cb93a7ba759d6b9e3c0cb"), "Rollno" : 2, "Name" : "Sachin", "Marks" : 78 } > db.Student.find({$or:[{Rollno:1}, {Rollno:3}]}).pretty() { "\_id" : ObjectId("637cb8fd7ba759d6b9e3c0ca"), "Rollno" : 1, "Name" : "Ram", "Marks" : 90 } > db.Stuent.find({$or:[{Rollno:1},{Name: 'Sachin'}]}).pretty() > db.Stuent.find({$or:[{Rollno:2},{Name: 'Sachin'}]}).pretty() > db.Student.find({$or:[{Rollno:1},{Name: 'Sachin'}]}).pretty() { "\_id" : ObjectId("637cb8fd7ba759d6b9e3c0ca"), "Rollno" : 1, "Name" : "Ram", "Marks" : 90 } { "\_id" : ObjectId("637cb93a7ba759d6b9e3c0cb"), "Rollno" : 2, "Name" : "Sachin", "Marks" : 78 } > db.Student.find({Rollno:1, $or:[{Marks:{$gt:85}}]}).pretty() { "\_id" : ObjectId("637cb8fd7ba759d6b9e3c0ca"), "Rollno" : 1, "Name" : "Ram", "Marks" : 90 } > db.Student.update({Rollno:1},{$set:{Marks:88}}) WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 }) > db.Student.finD().pretty(); 2022-11-22T17:34:57.777+0530 TypeError: Property 'finD' of object Student1.Student is not a function > db.Student.find().pretty(); { "\_id" : ObjectId("637cb8fd7ba759d6b9e3c0ca"), "Rollno" : 1, "Name" : "Ram", "Marks" : 88 } { "\_id" : ObjectId("637cb93a7ba759d6b9e3c0cb"), "Rollno" : 2, "Name" : "Sachin", "Marks" : 78 } > db.Student.remove({Rollno:2}) WriteResult({ "nRemoved" : 1 }) > db.Student.find().pretty(); { "\_id" : ObjectId("637cb8fd7ba759d6b9e3c0ca"), "Rollno" : 1, "Name" : "Ram", "Marks" : 88 } > db.Student.find().limit(1); { "\_id" : ObjectId("637cb8fd7ba759d6b9e3c0ca"), "Rollno" : 1, "Name" : "Ram", "Marks" : 88 } > db.Student.find({},{'Name':1,\_id:0}).limit(1); { "Name" : "Ram" } > db.STudent.insert({Rollno:2, Name:'Sachin', Marks:86}) WriteResult({ "nInserted" : 1 }) > db.Student.find() { "\_id" : ObjectId("637cb8fd7ba759d6b9e3c0ca"), "Rollno" : 1, "Name" : "Ram", "Marks" : 88 } > db.Student.insert({Rollno:2, Name:'Sachin', Marks:86}) WriteResult({ "nInserted" : 1 }) > db.Student.find() { "\_id" : ObjectId("637cb8fd7ba759d6b9e3c0ca"), "Rollno" : 1, "Name" : "Ram", "Marks" : 88 } { "\_id" : ObjectId("637cbbc87ba759d6b9e3c0cd"), "Rollno" : 2, "Name" : "Sachin", "Marks" : 86 } > db.Student.find().sort({'Marks':-1}) { "\_id" : ObjectId("637cb8fd7ba759d6b9e3c0ca"), "Rollno" : 1, "Name" : "Ram", "Marks" : 88 } { "\_id" : ObjectId("637cbbc87ba759d6b9e3c0cd"), "Rollno" : 2, "Name" : "Sachin", "Marks" : 86 } > db.Student.find().sort({'Marks':1}) { "\_id" : ObjectId("637cbbc87ba759d6b9e3c0cd"), "Rollno" : 2, "Name" : "Sachin", "Marks" : 86 } { "\_id" : ObjectId("637cb8fd7ba759d6b9e3c0ca"), "Rollno" : 1, "Name" : "Ram", "Marks" : 88 } >**

Q.9 Solve Following Queries on MongoDB

Consider the following

Student (rollno, name, marks)

Create student collection.

1. Insert at least 5 records with meaningful data.
2. Display all students. **db.Student.find()**
3. Find the students having marks >70. **db.Student.find({marks:{$gt:70}});**
4. Display students name having marks between 80 and 90. **db.Students.find({marks:{$gt:80,$lt:90}});**
5. Update the marks of “Ram” to 90. **db.Student.update({name:"Ram"},{$set:{marks:90}});**
6. Delete the students having marks = 75. **db.Student.delete({marks:75});**
7. Display top 2 students having highest marks. **db.Student.find().sort({marks:-1}).limit(2);**
8. Increment all student marks by 5%. **db.Student.update({},{$mul:{marks:1.05}},{multi:true});**
9. Display student name having highest marks. **db.Student.find().sort({marks:-1}).limit(1);**

Q.10 Solve Following Queries on MongoDB

Consider the following

Account (accno, branchname, balance)

Create account collection.

1. Insert at least 6 records with meaningful data.
2. Display all accounts. **db.Account.find().pretty();**
3. Find the accounts having balance >25000. **db.Account.find({balance:{$gt:25000}});**
4. Display account numbers having balance between 25000 and 35000. **db.Account.find({balance:{$gt:25000,$lt:35000}});**
5. Update the balance of accno. 101 to 35000. **db.Account.update({accno:101},{$set:{balance:35000}});**
6. Delete the accounts having balance 15000. **db.Account.remove({balance:15000});**
7. Display top 2 accounts having highest balance. **db.Account.find().sort({balance:-1}).limit(2);**
8. Display total balance for each branch. **db.Account**.
9. Display branch name having balance 20000. **db.Account.find({balance:20000},{branchname:1});**

Q.11 Consider the following collection

Student (rollno, name, marks, deptname, city) **db.createCollection("Students");**

Solve following Queries using MongoDB

* + 1. Display document of roll no. 1 **db.Students.find({rollno:1});**
    2. Display marks of “Ram”. **db.Students.find({name:”Ram”},{marks:1});**
    3. Display student having highest marks. **db.Students.find().sort({marks:-1}).limit(1);**
    4. Display students having marks greater than 70. **db.Students.find({marks:{$gt:70}});**
    5. Modify city of “Krishna” to “Pune”. **db.Students.update({name:”Krishna”},{$set:{city:”Pune”}});**
    6. Display maximum marks of each department. **db.student.aggregate([{$group:{\_id:"$dept",MaximumMarks:{$max:"$Marks"}}}]);**
    7. Display city of students having roll no. 10 to 20. **db.Students.find({rollno:{$gt:10,$lt:20}},{city:1});**
    8. Display student collection in descending order of marks. **db.student.find().sort({Marks:-1});**
    9. Delete document of roll no. 5. **db.student.remove({rollno:5});**

Q.12 Consider the following collection

Employee (empid, name, salary, deptname) **db.createCollection("employee");**

**db.employee.insertMany([**

**{"empid":1,"name":"ravi","salary":10000,"deptname":"cse"},**

**{"empid":2,"name":"riya","salary":56700,"deptname":"etc"},**

**{"empid":3,"name":"vikas","salary":98000,"deptname":"civil"},**

**{"empid":4,"name":"saurabh","salary":100900,"deptname":"mech"},**

**{"empid":5,"name":"guru","salary":7600,"deptname":"cse"}**

**]);**

Solve following Queries using MongoDB

1. Find employees of CSE departname. **db.employee.find({deptname:”CSE”},{name:1});**
2. Find the name of employees having salary 20000. **db.employee.find({salary:20000},{name:1});**
3. Find salary of ‘Ram” **db.employee.find({name:"Ram"},{salary:1});**
4. Find department name of ‘Krishna” **db.employee.find({name:”krishna”},{deptname:1});**
5. Update department of Krishna to ETC. **db.employee.update({name:”Krishna”},{$set:{deptname:”ETC”}});**
6. Increment salary by 1000 **db.employee.update({},{$inc:{salary:1000}},{multi:true});**
7. Delete document of “Ram” **db.employee.remove({name:”Ram”});**
8. Find total employees of each department. **db.employee.aggregate([{$group:{\_id:'$dept',deptcnt:{$sum:1}}}]);**

**MongoDB Queries**

**Create db Employee**

**> use cse**

**switched to db cse**

**> db.createCollection("Employee")**

**{ "ok" : 1 }**

**> show collections**

**Employee**

**Insert 10 employee documents**

**>**

**db.Employee.insert({Firstname:"Aditi",Lastname:"Bhogulkar",gender:"F",salary:30000,deptname:"**

**CSE"})**

**WriteResult({ "nInserted" : 1 })**

**>**

**db.Employee.insert({Firstname:"Adwait",Lastname:"Jankar",gender:"M",salary:25000,deptname:"**

**CSE"})**

**WriteResult({ "nInserted" : 1 })**

**>**

**db.Employee.insert({Firstname:"Tejashree",Lastname:"Khot",gender:"F",salary:25000,deptname:"**

**Mechanical"})**

**WriteResult({ "nInserted" : 1 })**

**>**

**db.Employee.insert({Firstname:"Tejas",Lastname:"Birje",gender:"M",salary:30000,deptname:"Rob**

**otics"})**

**WriteResult({ "nInserted" : 1 })**

**>**

**db.Employee.insert({Firstname:"Abhishek",Lastname:"Jagnade",gender:"M",salary:35000,deptnam**

**e:"ENTC"})**

**WriteResult({ "nInserted" : 1 })**

**>**

**db.Employee.insert({Firstname:"Ashutosh",Lastname:"Patil",gender:"M",salary:15000,deptname:"**

**CSE"})**

**WriteResult({ "nInserted" : 1 })**

**>**

**db.Employee.insert({Firstname:"Sakshi",Lastname:"Hasilkar",gender:"F",salary:35000,deptname:"**

**CSE"})**

**WriteResult({ "nInserted" : 1 })**

**>**

**db.Employee.insert({Firstname:"Rajat",Lastname:"Hasilkar",gender:"M",salary:45000,deptname:"**

**Mech"})**

**WriteResult({ "nInserted" : 1 })**

**>**

**db.Employee.insert({Firstname:"Shreya",Lastname:"Maske",gender:"F",salary:45000,deptname:"M**

**ech"})**

**WriteResult({ "nInserted" : 1 })>**

**db.Employee.insert({Firstname:"Geetanjali",Lastname:"Sultane",gender:"F",salary:35000,deptnam**

**e:"CSE"})**

**WriteResult({ "nInserted" : 1 })**

**> db.Employee.find()**

**{ "\_id" : ObjectId("638dde0d5b9cae48f3c90016"), "Firstname" : "Aditi", "Lastname" :**

**"Bhogulkar", "gender" : "F", "salary" : 30000, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638dde535b9cae48f3c90017"), "Firstname" : "Adwait", "Lastname" : "Jankar",**

**"gender" : "M", "salary" : 25000, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638ddef25b9cae48f3c9001a"), "Firstname" : "Tejashree", "Lastname" : "Khot",**

**"gender" : "F", "salary" : 25000, "deptname" : "Mechanical" }**

**{ "\_id" : ObjectId("638ddf405b9cae48f3c9001b"), "Firstname" : "Tejas", "Lastname" : "Birje",**

**"gender" : "M", "salary" : 30000, "deptname" : "Robotics" }**

**{ "\_id" : ObjectId("638de1655b9cae48f3c9001c"), "Firstname" : "Abhishek", "Lastname" :**

**"Jagnade", "gender" : "M", "salary" : 35000, "deptname" : "ENTC" }**

**{ "\_id" : ObjectId("638de1c05b9cae48f3c9001d"), "Firstname" : "Ashutosh", "Lastname" : "Patil",**

**"gender" : "M", "salary" : 15000, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638de1e75b9cae48f3c9001e"), "Firstname" : "Sakshi", "Lastname" : "Hasilkar",**

**"gender" : "F", "salary" : 35000, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638de2245b9cae48f3c9001f"), "Firstname" : "Rajat", "Lastname" : "Hasilkar",**

**"gender" : "M", "salary" : 45000, "deptname" : "Mech" }**

**{ "\_id" : ObjectId("638de2685b9cae48f3c90021"), "Firstname" : "Shreya", "Lastname" : "Maske",**

**"gender" : "F", "salary" : 45000, "deptname" : "Mech" }**

**{ "\_id" : ObjectId("638de29e5b9cae48f3c90022"), "Firstname" : "Geetanjali", "Lastname" :**

**"Sultane", "gender" : "F", "salary" : 35000, "deptname" : "CSE" }**

**Salary greater than 20000**

**> db.Employee.find({salary:{$gt:20000}});**

**{ "\_id" : ObjectId("638dde0d5b9cae48f3c90016"), "Firstname" : "Aditi", "Lastname" :**

**"Bhogulkar", "gender" : "F", "salary" : 30000, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638dde535b9cae48f3c90017"), "Firstname" : "Adwait", "Lastname" : "Jankar",**

**"gender" : "M", "salary" : 25000, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638ddef25b9cae48f3c9001a"), "Firstname" : "Tejashree", "Lastname" : "Khot",**

**"gender" : "F", "salary" : 25000, "deptname" : "Mechanical" }**

**{ "\_id" : ObjectId("638ddf405b9cae48f3c9001b"), "Firstname" : "Tejas", "Lastname" : "Birje",**

**"gender" : "M", "salary" : 30000, "deptname" : "Robotics" }**

**{ "\_id" : ObjectId("638de1655b9cae48f3c9001c"), "Firstname" : "Abhishek", "Lastname" :**

**"Jagnade", "gender" : "M", "salary" : 35000, "deptname" : "ENTC" }**

**{ "\_id" : ObjectId("638de1e75b9cae48f3c9001e"), "Firstname" : "Sakshi", "Lastname" : "Hasilkar",**

**"gender" : "F", "salary" : 35000, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638de2245b9cae48f3c9001f"), "Firstname" : "Rajat", "Lastname" : "Hasilkar",**

**"gender" : "M", "salary" : 45000, "deptname" : "Mech" }**

**{ "\_id" : ObjectId("638de2685b9cae48f3c90021"), "Firstname" : "Shreya", "Lastname" : "Maske",**

**"gender" : "F", "salary" : 45000, "deptname" : "Mech" }**

**{ "\_id" : ObjectId("638de29e5b9cae48f3c90022"), "Firstname" : "Geetanjali", "Lastname" :**

**"Sultane", "gender" : "F", "salary" : 35000, "deptname" : "CSE" }**

**Salary less than 20000**

**> db.Employee.find({salary:{$lt:20000}});**

**{ "\_id" : ObjectId("638de1c05b9cae48f3c9001d"), "Firstname" : "Ashutosh", "Lastname" : "Patil",**

**"gender" : "M", "salary" : 15000, "deptname" : "CSE" }**

**Employee of CSE DEPT**

**> db.Employee.find({deptname:'CSE'});**

**{ "\_id" : ObjectId("638dde0d5b9cae48f3c90016"), "Firstname" : "Aditi", "Lastname" :**

**"Bhogulkar", "gender" : "F", "salary" : 30000, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638dde535b9cae48f3c90017"), "Firstname" : "Adwait", "Lastname" : "Jankar",**

**"gender" : "M", "salary" : 25000, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638de1c05b9cae48f3c9001d"), "Firstname" : "Ashutosh", "Lastname" : "Patil",**

**"gender" : "M", "salary" : 15000, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638de1e75b9cae48f3c9001e"), "Firstname" : "Sakshi", "Lastname" : "Hasilkar",**

**"gender" : "F", "salary" : 35000, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638de29e5b9cae48f3c90022"), "Firstname" : "Geetanjali", "Lastname" :**

**"Sultane", "gender" : "F", "salary" : 35000, "deptname" : "CSE" }**

**Salary of Shreya**

**> db.Employee.find({Firstname:'Shreya'},{salary:1});**

**{ "\_id" : ObjectId("638de2685b9cae48f3c90021"), "salary" : 45000 }**

**Max salary of dept**

**> db.Employee.aggregate([{$group:{\_id:'$deptname',Maximumsalary:{$max:'$salary'}}}]);**

**{ "\_id" : "Mech", "Maximumsalary" : 45000 }**

**{ "\_id" : "CSE", "Maximumsalary" : 35000 }**

**{ "\_id" : "Robotics", "Maximumsalary" : 30000 }**

**{ "\_id" : "ENTC", "Maximumsalary" : 35000 }**

**{ "\_id" : "Mechanical", "Maximumsalary" : 25000 }**

**Update Salary by 10%**

**> db.Employee.update({},{$mul:{salary:1.1}},{multi:true});**

**WriteResult({ "nMatched" : 10, "nUpserted" : 0, "nModified" : 10 })**

**> db.Employee.find();**

**{ "\_id" : ObjectId("638dde0d5b9cae48f3c90016"), "Firstname" : "Aditi", "Lastname" :**

**"Bhogulkar", "gender" : "F", "salary" : 33000, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638dde535b9cae48f3c90017"), "Firstname" : "Adwait", "Lastname" : "Jankar",**

**"gender" : "M", "salary" : 27500.000000000004, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638ddef25b9cae48f3c9001a"), "Firstname" : "Tejashree", "Lastname" : "Khot",**

**"gender" : "F", "salary" : 27500.000000000004, "deptname" : "Mechanical" }**

**{ "\_id" : ObjectId("638ddf405b9cae48f3c9001b"), "Firstname" : "Tejas", "Lastname" : "Birje",**

**"gender" : "M", "salary" : 33000, "deptname" : "Robotics" }**

**{ "\_id" : ObjectId("638de1655b9cae48f3c9001c"), "Firstname" : "Abhishek", "Lastname" :**

**"Jagnade", "gender" : "M", "salary" : 38500, "deptname" : "ENTC" }**

**{ "\_id" : ObjectId("638de1c05b9cae48f3c9001d"), "Firstname" : "Ashutosh", "Lastname" : "Patil",**

**"gender" : "M", "salary" : 16500, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638de1e75b9cae48f3c9001e"), "Firstname" : "Sakshi", "Lastname" : "Hasilkar",**

**"gender" : "F", "salary" : 38500, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638de2245b9cae48f3c9001f"), "Firstname" : "Rajat", "Lastname" : "Hasilkar",**

**"gender" : "M", "salary" : 49500.00000000001, "deptname" : "Mech" }**

**{ "\_id" : ObjectId("638de2685b9cae48f3c90021"), "Firstname" : "Shreya", "Lastname" : "Maske",**

**"gender" : "F", "salary" : 49500.00000000001, "deptname" : "Mech" }**

**{ "\_id" : ObjectId("638de29e5b9cae48f3c90022"), "Firstname" : "Geetanjali", "Lastname" :**

**"Sultane", "gender" : "F", "salary" : 38500, "deptname" : "CSE" }**

**Update deptname of shreya to CSE**

**> db.Employee.update({Firstname:"Shreya"},{$set:{deptname:'CSE'}});**

**WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })**

**> db.Employee.find();**

**{ "\_id" : ObjectId("638dde0d5b9cae48f3c90016"), "Firstname" : "Aditi", "Lastname" :**

**"Bhogulkar", "gender" : "F", "salary" : 33000, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638dde535b9cae48f3c90017"), "Firstname" : "Adwait", "Lastname" : "Jankar",**

**"gender" : "M", "salary" : 27500.000000000004, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638ddef25b9cae48f3c9001a"), "Firstname" : "Tejashree", "Lastname" : "Khot",**

**"gender" : "F", "salary" : 27500.000000000004, "deptname" : "Mechanical" }**

**{ "\_id" : ObjectId("638ddf405b9cae48f3c9001b"), "Firstname" : "Tejas", "Lastname" : "Birje",**

**"gender" : "M", "salary" : 33000, "deptname" : "Robotics" }**

**{ "\_id" : ObjectId("638de1655b9cae48f3c9001c"), "Firstname" : "Abhishek", "Lastname" :**

**"Jagnade", "gender" : "M", "salary" : 38500, "deptname" : "ENTC" }**

**{ "\_id" : ObjectId("638de1c05b9cae48f3c9001d"), "Firstname" : "Ashutosh", "Lastname" : "Patil",**

**"gender" : "M", "salary" : 16500, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638de1e75b9cae48f3c9001e"), "Firstname" : "Sakshi", "Lastname" : "Hasilkar",**

**"gender" : "F", "salary" : 38500, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638de2245b9cae48f3c9001f"), "Firstname" : "Rajat", "Lastname" : "Hasilkar",**

**"gender" : "M", "salary" : 49500.00000000001, "deptname" : "Mech" }**

**{ "\_id" : ObjectId("638de2685b9cae48f3c90021"), "Firstname" : "Shreya", "Lastname" : "Maske",**

**"gender" : "F", "salary" : 49500.00000000001, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638de29e5b9cae48f3c90022"), "Firstname" : "Geetanjali", "Lastname" :**

**"Sultane", "gender" : "F", "salary" : 38500, "deptname" : "CSE" }**

**Display employees salary in descending order**

**> db.Employee.find({},{Firstname:1,salary:1,\_id:0}).sort({salary:-1});**

**{ "Firstname" : "Rajat", "salary" : 49500.00000000001 }**

**{ "Firstname" : "Shreya", "salary" : 49500.00000000001 }**

**{ "Firstname" : "Abhishek", "salary" : 38500 }**

**{ "Firstname" : "Sakshi", "salary" : 38500 }**

**{ "Firstname" : "Geetanjali", "salary" : 38500 }**

**{ "Firstname" : "Aditi", "salary" : 33000 }**

**{ "Firstname" : "Tejas", "salary" : 33000 }**

**{ "Firstname" : "Adwait", "salary" : 27500.000000000004 }**

**{ "Firstname" : "Tejashree", "salary" : 27500.000000000004 }**

**{ "Firstname" : "Ashutosh", "salary" : 16500 }**

**Find employee whose salary is 33000**

**> db.Employee.find({salary:{$eq:33000}});**

**{ "\_id" : ObjectId("638dde0d5b9cae48f3c90016"), "Firstname" : "Aditi", "Lastname" :**

**"Bhogulkar", "gender" : "F", "salary" : 33000, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638ddf405b9cae48f3c9001b"), "Firstname" : "Tejas", "Lastname" : "Birje",**

**"gender" : "M", "salary" : 33000, "deptname" : "Robotics" }**

**Find top 5 employees having highest salary**

**> db.Employee.find({},{Firstname:1,salary:1,\_id:0}).sort({salary:-1}).limit(5);**

**{ "Firstname" : "Shreya", "salary" : 49500.00000000001 }**

**{ "Firstname" : "Rajat", "salary" : 49500.00000000001 }**

**{ "Firstname" : "Geetanjali", "salary" : 38500 }**

**{ "Firstname" : "Abhishek", "salary" : 38500 }**

**{ "Firstname" : "Sakshi", "salary" : 38500 }**

**Delete document of adwait**

**> db.Employee.remove({Firstname:"Adwait"})**

**WriteResult({ "nRemoved" : 1 })**

**> db.Employee.find()**

**{ "\_id" : ObjectId("638dde0d5b9cae48f3c90016"), "Firstname" : "Aditi", "Lastname" :**

**"Bhogulkar", "gender" : "F", "salary" : 33000, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638ddef25b9cae48f3c9001a"), "Firstname" : "Tejashree", "Lastname" : "Khot",**

**"gender" : "F", "salary" : 27500.000000000004, "deptname" : "Mechanical" }**

**{ "\_id" : ObjectId("638ddf405b9cae48f3c9001b"), "Firstname" : "Tejas", "Lastname" : "Birje",**

**"gender" : "M", "salary" : 33000, "deptname" : "Robotics" }**

**{ "\_id" : ObjectId("638de1655b9cae48f3c9001c"), "Firstname" : "Abhishek", "Lastname" :**

**"Jagnade", "gender" : "M", "salary" : 38500, "deptname" : "ENTC" }**

**{ "\_id" : ObjectId("638de1c05b9cae48f3c9001d"), "Firstname" : "Ashutosh", "Lastname" : "Patil",**

**"gender" : "M", "salary" : 16500, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638de1e75b9cae48f3c9001e"), "Firstname" : "Sakshi", "Lastname" : "Hasilkar",**

**"gender" : "F", "salary" : 38500, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638de2245b9cae48f3c9001f"), "Firstname" : "Rajat", "Lastname" : "Hasilkar",**

**"gender" : "M", "salary" : 49500.00000000001, "deptname" : "Mech" }**

**{ "\_id" : ObjectId("638de2685b9cae48f3c90021"), "Firstname" : "Shreya", "Lastname" : "Maske",**

**"gender" : "F", "salary" : 49500.00000000001, "deptname" : "CSE" }**

**{ "\_id" : ObjectId("638de29e5b9cae48f3c90022"), "Firstname" : "Geetanjali", "Lastname" :**

**"Sultane", "gender" : "F", "salary" : 38500, "deptname" : "CSE" }**

**>**

Q.13 Consider the following collection

Employee (empid, name, salary, deptname)

Solve following Queries using MongoDB

1. Find the name of employees having salary 25000.
2. Display employees in descending order of salary.
3. Display top 2 employees having highest salary.
4. Find maximum salary.
5. Increment all employee salary by 6%.
6. Display employee name and salary
7. Delete documents of Chemical department.
8. Display salary and department name of “Ram”.

Q.14

a)

1. create table locations(locid varchar(5),city varchar(10),state varchar(5),country varchar(10));
2. create table sales(pid varchar(5),timeid varchar(5),locid varchar(5),sales varchar(5));
3. create table products(pid varchar(5), pname varchar(10), category varchar(15),price varchar(5));
4. create table time(timeid varchar(5),year varchar(5));

Insert appropriate data and perform OLAP Rollup and CUBE operators.

**SELECT T.year,L.state,SUM(S.sales) FROM sales S, time T, locations L WHERE S.timeid=T.timeid and S.locid=L.locid GROUP BY CUBE (T.year, L.state);**

**SELECT T.year,L.state,SUM(S.sales) FROM sales S, time T, locations L WHERE S.timeid=T.timeid and S.locid=L.locid GROUP BY ROLLUP (T.year, L.state);**

b) Demonstrate Grant and Revoke commands

**Title : Implementation of OLAP Queries.**

create table locations(locid varchar(5),city varchar(10),state varchar(5),country varchar(10));

insert into locations values(1,'madison','W1','USA');

insert into locations values(2,'fresno','CA','USA');

insert into locations values(5,'Chennai','TN','India');

select \* from locations;

create table sales(pid varchar(5),timeid varchar(5),locid varchar(5),sales varchar(5));

insert into sales values(12,3,1,50);

insert into sales values(11,1,2,35);

insert into sales values(11,2,2,22);

insert into sales values(11,3,2,10);

insert into sales values(12,1,2,26);

insert into sales values(12,2,2,45);

insert into sales values(12,3,2,20);

insert into sales values(13,1,2,20);

insert into sales values(13,2,2,40);

insert into sales values(13,3,2,5);

select \* from sales;

create table products(pid varchar(5), pname varchar(10), category varchar(15),price varchar(5));

insert into products values(11,'Lee Jeans','Apparell',25);

insert into products values(12,'Zord','Toys',18);

insert into products values(13,'Biro Pen','Stationary',2);

select \* from products;

create table time(timeid varchar(5),year varchar(5));

insert into time values(1,2016);

insert into time values(2,2017);

insert into time values(3,2018);

select \* from time;

SELECT T.year,L.state,SUM(S.sales)

FROM sales S, time T, locations L

WHERE S.timeid=T.timeid and S.locid=L.locid

GROUP BY T.year, L.state;

SELECT T.year,SUM(S.sales)

FROM sales S, time T

WHERE S.timeid=T.timeid

GROUP BY T.year;

SELECT L.state,SUM(S.sales)

FROM sales S, locations L

WHERE S.locid=L.locid

GROUP BY L.state;

SELECT T.year,L.state,SUM(S.sales)

FROM sales S, time T, locations L

WHERE S.timeid=T.timeid and S.locid=L.locid

GROUP BY CUBE (T.year, L.state);

SELECT T.year,L.state,SUM(S.sales)

FROM sales S, time T, locations L

WHERE S.timeid=T.timeid and S.locid=L.locid

GROUP BY ROLLUP (T.year, L.state);

Q.15 Solve Following Queries on MongoDB

Consider the following

Student (rollno, name, marks)

1. Display document of "Ram"
2. Marks greater than 80 and less than 90
3. Display name and marks of students having marks equal or greater than 80 and < 90
4. Display name and marks of student having rollno. 1
5. Display documents in ascending order of name
6. Display first 2 documents
7. Display document of Ram or rollno 3
8. Display document of Ram and rollno 1
9. Display all in documents in descending order of Marks
10. Display name and marks of student having highest marks

Q.16 Solve Following Queries on CouchDB

Consider the following

Employee (empid, name, salary, deptname)

1. Display document of "Ram" **{ “selector”:{“name”:”Ram”}}**
2. Display salary greater than 20000 and less than 30000 **{“selector”:{“salary”:{“$gt”:20000,”$lt”:30000}}}**
3. Display name and department name of employees having salary equal or greater than 10000 and < 15000 **{“selector”:{“salary”:{“$gte”:10000,”$lt”:15000}},”fields”:[“name”,”deptname”]}**
4. Display name and salary of employee having empid 1 **{“selector”:{“empid”:1},”fields”:[“name”,”salary”]}**
5. Display documents in ascending order of name **{“selector”:{},”Fields”:[“name”,”salary”],”sort”:[“name”]}**
6. Display first 2 documents **{“selector”:{},”Fields”:[“name”,”salary”],”limit”:2}**
7. Display document of Ram or empid. 3 **{“selector”:{“$or”:[{“name”:”Ram”},{“empid”:3}]}}**
8. Display all in documents in descending order of salary **{“selector”:{},”sort”:[{“salary”:desc}]}**
9. Display name and salary of employee having highest salary.

**{“selector”:{},”Fields”:[“name”,”salary”],”sort”:[{“salary”:desc}],”limit”:1}**

1. Display department having highest salary.

**Question Bank for Oral exam**

**Unit 1: Parallel and Distributed Databases**

1. What is a parallel system?

2. What is throughput and response time? Or how to measure performance of database systems?

3. What is Speedup and Scaleup? Explain linear and sublinear speedup and scaleup.

4. Explain parallel database architectures. Give their advantages and disadvantages.

5. What is distributed database system? Explain its advantages and disadvantages.

6. Explain different partitioning techniques used in parallel database systems.

7. Compare different partitioning techniques used in parallel database systems.

8. What is skew? How to handle it?

9. What is interquery and intraquery parallelism?

10. What is interoperation and intraoperation parallelism?

11. What is pipelined parallelism and independent parallelism?

12. Explain how to store data in distributed database system.

13. Explain replication, its types, and advantages, disadvantages.

14. Explain horizontal and vertical fragmentation.

15. Explain 2 phase commit (2PC) protocol.

16. How 2PC protocol handles failures?

17. Explain 3 phase commit (3PC) protocol.

18. Explain Semijoin in detail.

19. What is local transaction and global transaction?

20. What is transaction manager and transaction coordinator?

21. What are ACID properties?

**Unit 2: Advanced SQL**

1. What is different SQL character, number functions?

2. What is view, relation, materialized views, primary key, foreign key, unique, check and not null constraint?

3. What are different DDL and DML commands? Explain it with syntax and example.

4. What is PLSQL? Why it is used? Explain PLSQL block.

5. What is stored procedure? What are its advantages?

6. What is PLSQL function?

7. Give difference between Procedure and function.

8. What are in, out and inout parameters?

9. Write PLSQL procedure and function for factorial, prime number, reverse number, greater from 3 numbers.

10. What is trigger? Explain its types and applications?

11. What is cursor? Explain its types and attributes?

12. Define embedded and dynamic SQL?

**Unit 3: NoSQL Database Management**

1. What is NoSQL? Explain its features, advantages and disadvantages.

2. Explain types of NoSQL databases with example.

3. What is MongoDB? Explain its features, advantages and disadvantages.

4. Explain all MongoDB commands/functions used for creating collections, retrieving, updating,

deleting, sorting, limit, inserting, aggregating documents with examples.

5. What is CouchDB? Explain its features, advantages and disadvantages.

6. Give difference between MongoDB and CouchDB.

**Unit 4: Database Administration and Security**

1. What is need for and role of databases in an organization?

2. What is DBA? What are its responsibilities?

3. What is DA? What are its responsibilities?

4. Explain DBA’s managerial role.

5. Explain DBA’s technical role.

6. Explain database administration tools- Data dictionary and CASE tools.

7. Describe data dictionary.

8. Describe CASE tools.

9. What are security vulnerabilities and its measures?

10. Describe grant and revoke commands.

**Unit 5: Business Intelligence and Data Warehouse**

1. What is data, information, knowledge?

2. What is decision support system?

3. What are OLTP and OLAP?

4. What is data warehouse? Give its characteristics.

5. What is Data Mart? Give example.

6. Explain types of OLAP.

7. Explain Star Schema, Snowflake and fact constellation schema.

8. What is Fact table and dimension table?

9. Explain OLAP operations.

10. Explain materialized views.

11. Explain Rollup and Cube operator.

12. What is Business Intelligence? Draw Business Intelligence architecture.

**Unit 6: Data Mining**

1. What is Data Mining?

2. What are the different data mining tasks? Explain each of them.

3. What is KDD? Explain steps performed in KDD.

4. What are the different data mining issues?

5. What are the different data mining algorithms?

**Advanced Database Systems**

**Sample Program list**

Q.1 Write Java-MySQL program for horizontal fragmentation (client–server).

Q.2 Write Java-MySQL program for vertical fragmentation (client–server).

Q.3 Write Java-MySQL program for Semijoin.

Q.4 a) Write PLSQL Procedure to find given number is prime or not.

b) Write PLSQL Procedure to find factorial of given number.

Q.5 a) Write PLSQL Procedure to find reverse of given number.

b) Write PLSQL function to find factorial of given number.

Q.6 a) Write PLSQL function to find reverse of given String.

b) Write PLSQL procedure to find average of given numbers.

Q. 7 a) Write PLSQL procedure to find greater from three numbers.

b) Write a cursor that will increase salary by 10% if salary is > 10000 otherwise by 5%.

Q.8 Solve Following Queries on MongoDB

Consider the following

Employee (empid, name, salary, designation)

Create employee collection.

1. Insert at least 5 records with meaningful data.
2. Display all employees.
3. Find the employees whose salary is >10000.
4. Display the employee name whose having salary either 15000 or 25000
5. Update designation of employee 11 to ‘Asst. Prof.’
6. Delete the employees having designation “supervisor”.
7. Display top 2 employees having highest salary.
8. Increment all employee salary by 1000.
9. Increment all employee salary by 6%.

Q.9 Solve Following Queries on MongoDB

Consider the following

Student (rollno, name, marks)

Create student collection.

1. Insert at least 5 records with meaningful data.
2. Display all students.
3. Find the students having marks >70.
4. Display students name having marks between 80 and 90.
5. Update the marks of “Ram” to 90.
6. Delete the students having marks = 75.
7. Display top 2 students having highest marks.
8. Increment all student marks by 5%.
9. Display student name having highest marks.

Q.10 Solve Following Queries on MongoDB

Consider the following

Account (accno, branchname, balance)

Create account collection.

1. Insert at least 6 records with meaningful data.
2. Display all accounts.
3. Find the accounts having balance >25000.
4. Display account numbers having balance between 25000 and 35000.
5. Update the balance of accno. 101 to 35000.
6. Delete the accounts having balance 15000.
7. Display top 2 accounts having highest balance.
8. Display total balance for each branch.
9. Display branch name having balance 20000.

Q.11 Consider the following collection

Student (rollno, name, marks, deptname, city)

Solve following Queries using MongoDB

* + 1. Display document of roll no. 1
    2. Display marks of “Ram”.
    3. Display student having highest marks.
    4. Display students having marks greater than 70.
    5. Modify city of “Krishna” to “Pune”.
    6. Display maximum marks of each department.
    7. Display city of students having roll no. 10 to 20.
    8. Display student collection in descending order of marks.
    9. Delete document of roll no. 5.

Q.12 Consider the following collection

Employee (empid, name, salary, deptname)

Solve following Queries using MongoDB

1. Find employees of CSE departname.
2. Find the name of employees having salary 20000.
3. Find salary of ‘Ram”
4. Find department name of ‘Krishna”
5. Update department of Krishna to ETC.
6. Increment salary by 1000
7. Delete document of “Ram”
8. Find total employees of each department.

Q.13 Consider the following collection

Employee (empid, name, salary, deptname)

Solve following Queries using MongoDB

1. Find the name of employees having salary 25000.
2. Display employees in descending order of salary.
3. Display top 2 employees having highest salary.
4. Find maximum salary.
5. Increment all employee salary by 6%.
6. Display employee name and salary
7. Delete documents of Chemical department.
8. Display salary and department name of “Ram”.

Q.14

a)

1. create table locations(locid varchar(5),city varchar(10),state varchar(5),country varchar(10));
2. create table sales(pid varchar(5),timeid varchar(5),locid varchar(5),sales varchar(5));
3. create table products(pid varchar(5), pname varchar(10), category varchar(15),price varchar(5));
4. create table time(timeid varchar(5),year varchar(5));

Insert appropriate data and perform OLAP Rollup and CUBE operators.

b) Demonstrate Grant and Revoke commands

Q.15 Solve Following Queries on MongoDB

Consider the following

Student (rollno, name, marks)

1. Display document of "Ram"
2. Marks greater than 80 and less than 90
3. Display name and marks of students having marks equal or greater than 80 and < 90
4. Display name and marks of student having rollno. 1
5. Display documents in ascending order of name
6. Display first 2 documents
7. Display document of Ram or rollno 3
8. Display document of Ram and rollno 1
9. Display all in documents in descending order of Marks
10. Display name and marks of student having highest marks

Q.16 Solve Following Queries on CouchDB

Consider the following

Employee (empid, name, salary, deptname)

1. Display document of "Ram"
2. Display salary greater than 20000 and less than 30000
3. Display name and department name of employees having salary equal or greater than 10000 and < 15000
4. Display name and salary of employee having empid 1
5. Display documents in ascending order of name
6. Display first 2 documents
7. Display document of Ram or empid. 3
8. Display all in documents in descending order of salary
9. Display name and salary of employee having highest salary.
10. Display department having highest salary.

**Question Bank for Oral exam**

**Unit 1: Parallel and Distributed Databases**

1. What is a parallel system?

2. What is throughput and response time? Or how to measure performance of database systems?

3. What is Speedup and Scaleup? Explain linear and sublinear speedup and scaleup.

4. Explain parallel database architectures. Give their advantages and disadvantages.

5. What is distributed database system? Explain its advantages and disadvantages.

6. Explain different partitioning techniques used in parallel database systems.

7. Compare different partitioning techniques used in parallel database systems.

8. What is skew? How to handle it?

9. What is interquery and intraquery parallelism?

10. What is interoperation and intraoperation parallelism?

11. What is pipelined parallelism and independent parallelism?

12. Explain how to store data in distributed database system.

13. Explain replication, its types, and advantages, disadvantages.

14. Explain horizontal and vertical fragmentation.

15. Explain 2 phase commit (2PC) protocol.

16. How 2PC protocol handles failures?

17. Explain 3 phase commit (3PC) protocol.

18. Explain Semijoin in detail.

19. What is local transaction and global transaction?

20. What is transaction manager and transaction coordinator?

21. What are ACID properties?

**Unit 2: Advanced SQL**

1. What is different SQL character, number functions?

2. What is view, relation, materialized views, primary key, foreign key, unique, check and not null constraint?

3. What are different DDL and DML commands? Explain it with syntax and example.

4. What is PLSQL? Why it is used? Explain PLSQL block.

5. What is stored procedure? What are its advantages?

6. What is PLSQL function?

7. Give difference between Procedure and function.

8. What are in, out and inout parameters?

9. Write PLSQL procedure and function for factorial, prime number, reverse number, greater from 3 numbers.

10. What is trigger? Explain its types and applications?

11. What is cursor? Explain its types and attributes?

12. Define embedded and dynamic SQL?

**Unit 3: NoSQL Database Management**

1. What is NoSQL? Explain its features, advantages and disadvantages.

2. Explain types of NoSQL databases with example.

3. What is MongoDB? Explain its features, advantages and disadvantages.

4. Explain all MongoDB commands/functions used for creating collections, retrieving, updating,

deleting, sorting, limit, inserting, aggregating documents with examples.

5. What is CouchDB? Explain its features, advantages and disadvantages.

6. Give difference between MongoDB and CouchDB.

**Unit 4: Database Administration and Security**

1. What is need for and role of databases in an organization?

2. What is DBA? What are its responsibilities?

3. What is DA? What are its responsibilities?

4. Explain DBA’s managerial role.

5. Explain DBA’s technical role.

6. Explain database administration tools- Data dictionary and CASE tools.

7. Describe data dictionary.

8. Describe CASE tools.

9. What are security vulnerabilities and its measures?

10. Describe grant and revoke commands.

**Unit 5: Business Intelligence and Data Warehouse**

1. What is data, information, knowledge?

2. What is decision support system?

3. What are OLTP and OLAP?

4. What is data warehouse? Give its characteristics.

5. What is Data Mart? Give example.

6. Explain types of OLAP.

7. Explain Star Schema, Snowflake and fact constellation schema.

8. What is Fact table and dimension table?

9. Explain OLAP operations.

10. Explain materialized views.

11. Explain Rollup and Cube operator.

12. What is Business Intelligence? Draw Business Intelligence architecture.

**Unit 6: Data Mining**

1. What is Data Mining?

2. What are the different data mining tasks? Explain each of them.

3. What is KDD? Explain steps performed in KDD.

4. What are the different data mining issues?

5. What are the different data mining algorithms?

**Assignment No. 8**

Java-MongoDB Connectivity and CRUD Operations in MongoDB using Java

Just click on the following link to download the required driver:

<https://search.maven.org/remotecontent?filepath=org/mongodb/mongo-java-driver/2.12.3/mongo-java-driver-2.12.3.jar>

To export the required driver’s packages and classes in program:

**export CLASSPATH=$CLASSPATH:/home/ubuntu/Downloads/mongo-java-driver-2.12.3.jar**

**Program:**

import com.mongodb.BasicDBObject;

import com.mongodb.DB;

import com.mongodb.DBCollection;

import com.mongodb.DBCursor;

import com.mongodb.DBObject;

import com.mongodb.MongoClient;

import com.mongodb.WriteResult;

public class db1 {

public static void main(String[] args) throws Exception {

try{

MongoClient mongo = new MongoClient("localhost" , 27017);

DB db = mongo.getDB("dypcet");

//Create and Insert Operations

System.out.println("---Create and Insert Operations---\n");

System.out.println("Collection created successfully.");

DBCollection col=db.getCollection("student");

BasicDBObject b1 = new BasicDBObject();

b1.put("first\_name", "Sohail");

b1.put("last\_name","Shaikh");

b1.put("gender","Male");

b1.put("department","CSE");

col.insert(b1);

BasicDBObject b2 = new BasicDBObject();

b2.put("first\_name", "Vicky");

b2.put("last\_name","Singh");

b2.put("gender","Male");

b2.put("department","ENTC");

col.insert(b2);

BasicDBObject b3 = new BasicDBObject();

b3.put("first\_name", "Anuj");

b3.put("last\_name","Patil");

b3.put("gender","Male");

b3.put("department","Civil");

col.insert(b3);

System.out.println("Document inserted Successfully.\n");

//Retrieve Operation

System.out.println("---Retrieve Operation---\n");

DBCursor cursor = col.find();

while (cursor.hasNext())

{

System.out.println(cursor.next());

}

//Update Operation

System.out.println("\n---Update Operation---\n");

BasicDBObject query = new BasicDBObject();

query.put("department", "Civil");

BasicDBObject newDocument = new BasicDBObject();

newDocument.put("department", "Chemical");

BasicDBObject updateObject = new BasicDBObject();

updateObject.put("$set", newDocument);

db.getCollection("student").update(query, updateObject);

System.out.println("Document Updated Successfully.\n");

DBCursor cursor1 = col.find();

while (cursor1.hasNext())

{

System.out.println(cursor1.next());

}

//Delete Operation

System.out.println("\n---Delete Operation---\n");

BasicDBObject query1 = new BasicDBObject();

query1.put("first\_name", "Anuj");

db.getCollection("student").remove(query1);

System.out.println("Document Deleted Successfully.\n");

DBCursor cursor2 = col.find();

while (cursor2.hasNext())

{

System.out.println(cursor2.next());

}

}

catch(Exception e)

{

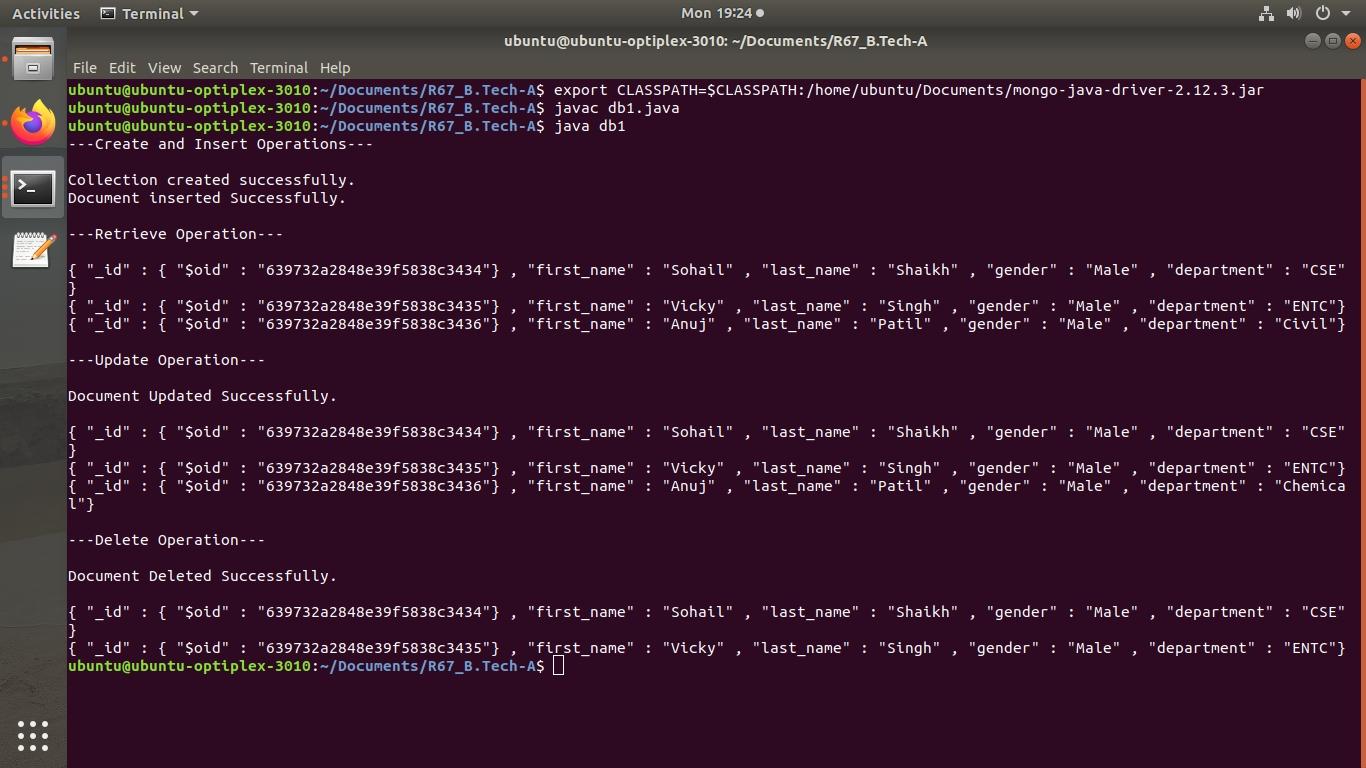
System.out.println(e);

}

}

}

**Output:**

****

**Assignment 4: Oracle Sequence and Synonym**

**-- Sequence Example**

SQL> create sequence productid start with 1 nocache;

Sequence created.

SQL> create sequence billno start with 1 nocache;

Sequence created.

SQL> create table product (productid number(5), Prodname varchar2(10), Prize number(5));

Table created.

SQL> insert into product values( productid.nextval, 'Pen', 10);

1 row created.

SQL> insert into product values( productid.nextval, 'Pencil',5);

1 row created.

SQL> select \* from product;

PRODUCTID PRODNAME PRIZE

---------- ---------- ----------

1 Pen 10

2 Pencil 5

SQL> insert into product values( productid.currval, 'notebook',50);

1 row created.

SQL> select \* from product;

PRODUCTID PRODNAME PRIZE

---------- ---------- ----------

1 Pen 10

2 Pencil 5

2 notebook 50

**--Synonym Example**

SQL> create table student(id number, roll\_no number, name varchar(10), dept varchar(10));

Table created.

SQL> insert into student values(1,82,'Shreya','CSE');

1 row created.

SQL> insert into student values(2,83,'Mayur','Civil');

1 row created.

SQL> insert into student values(3,84,'Atharva','ENTC');

1 row created.

SQL> create public synonym stud for student;

Synonym created.

SQL> select \* from stud;

ID ROLL\_NO NAME DEPT

---------- ---------- ---------- ----------

1 82 Shreya CSE

2 83 Mayur Civil

3 84 Atharva ENTC

SQL> select \* from stud where dept='CSE';

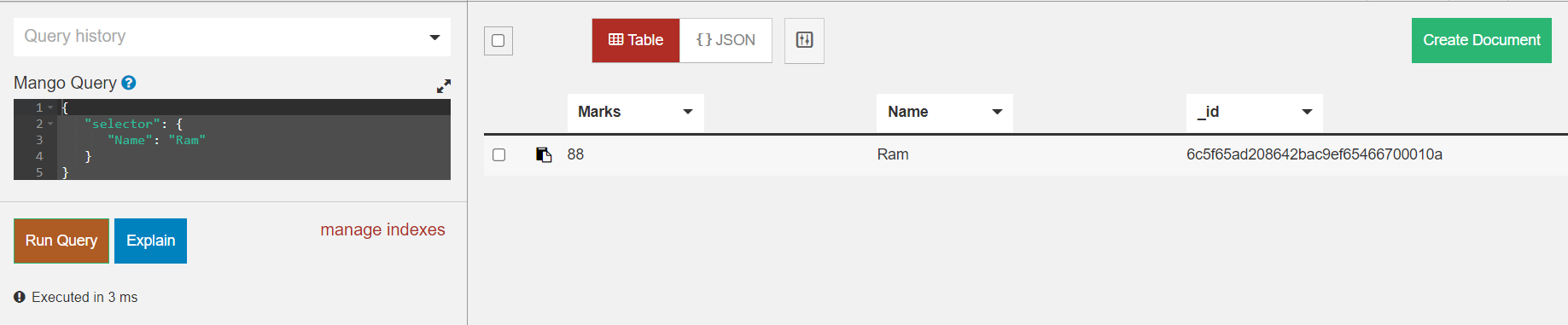
ID ROLL\_NO NAME DEPT

---------- ---------- ---------- ----------

1 82 Shreya CSE

Assignment 9: CouchDB Queries:

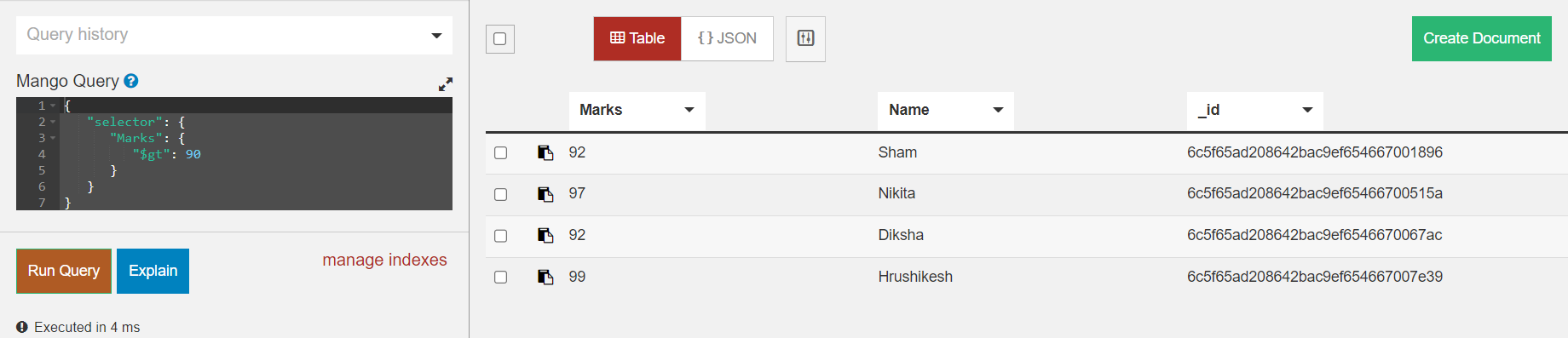
1. Display Document of Ram:



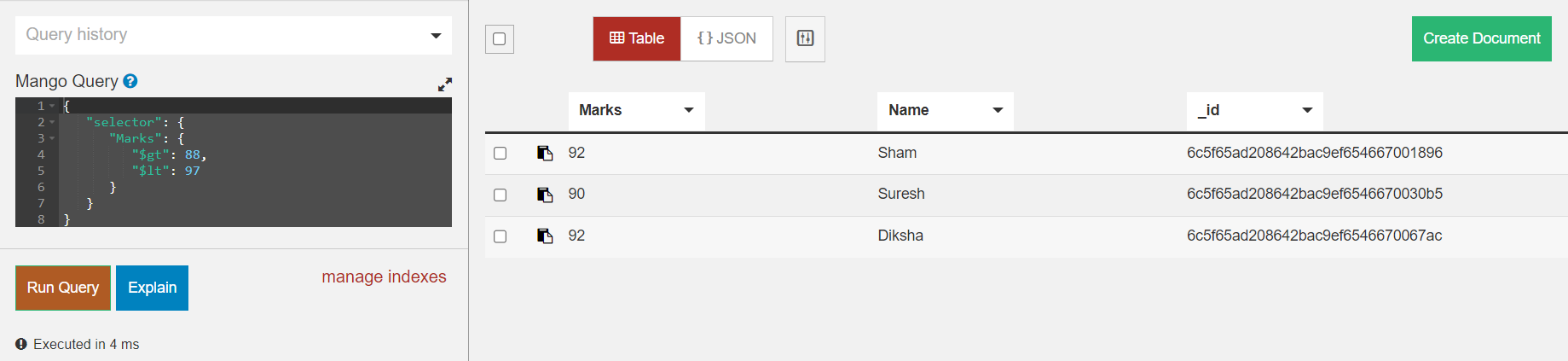
1. Marks Equal to 90:



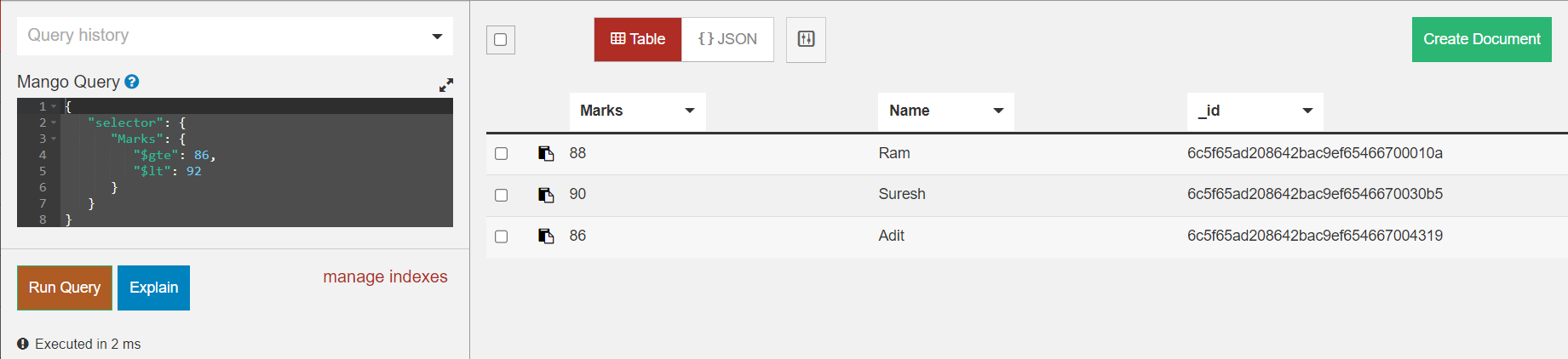
1. Marks greater than 90:



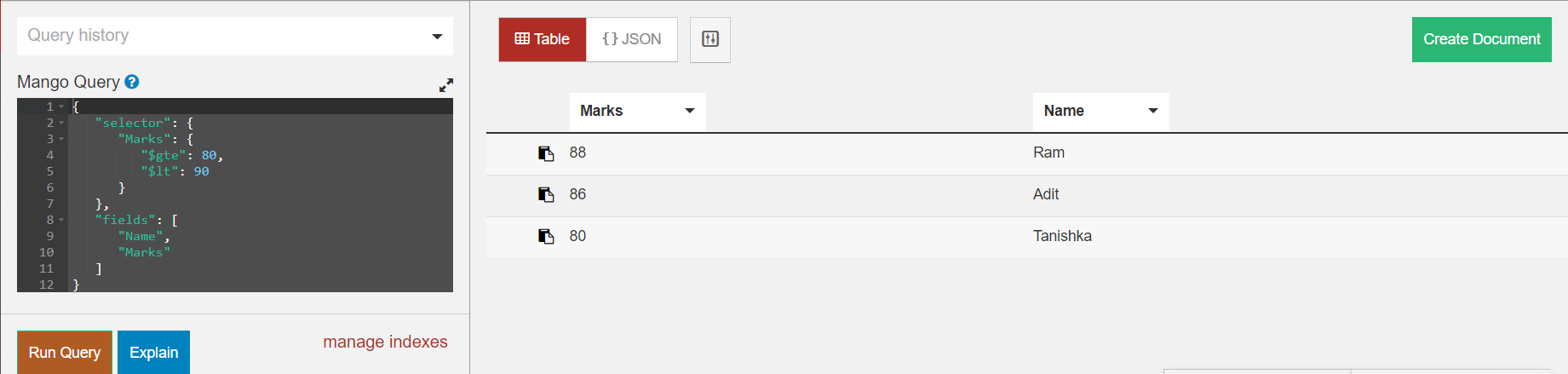
1. Marks greater than 88 and less than 97:



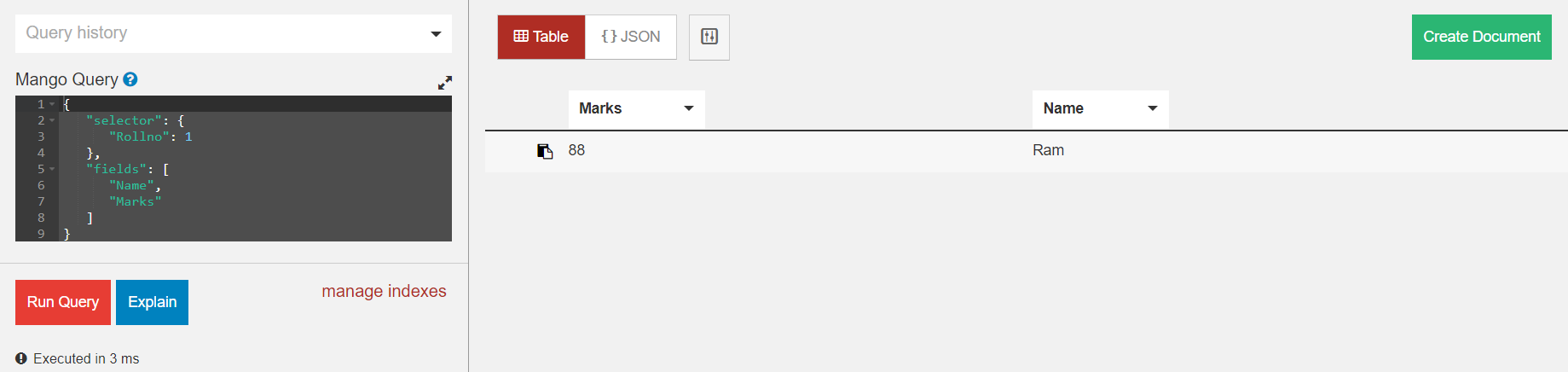
1. Marks equal to or greater than 86 and less than 92:



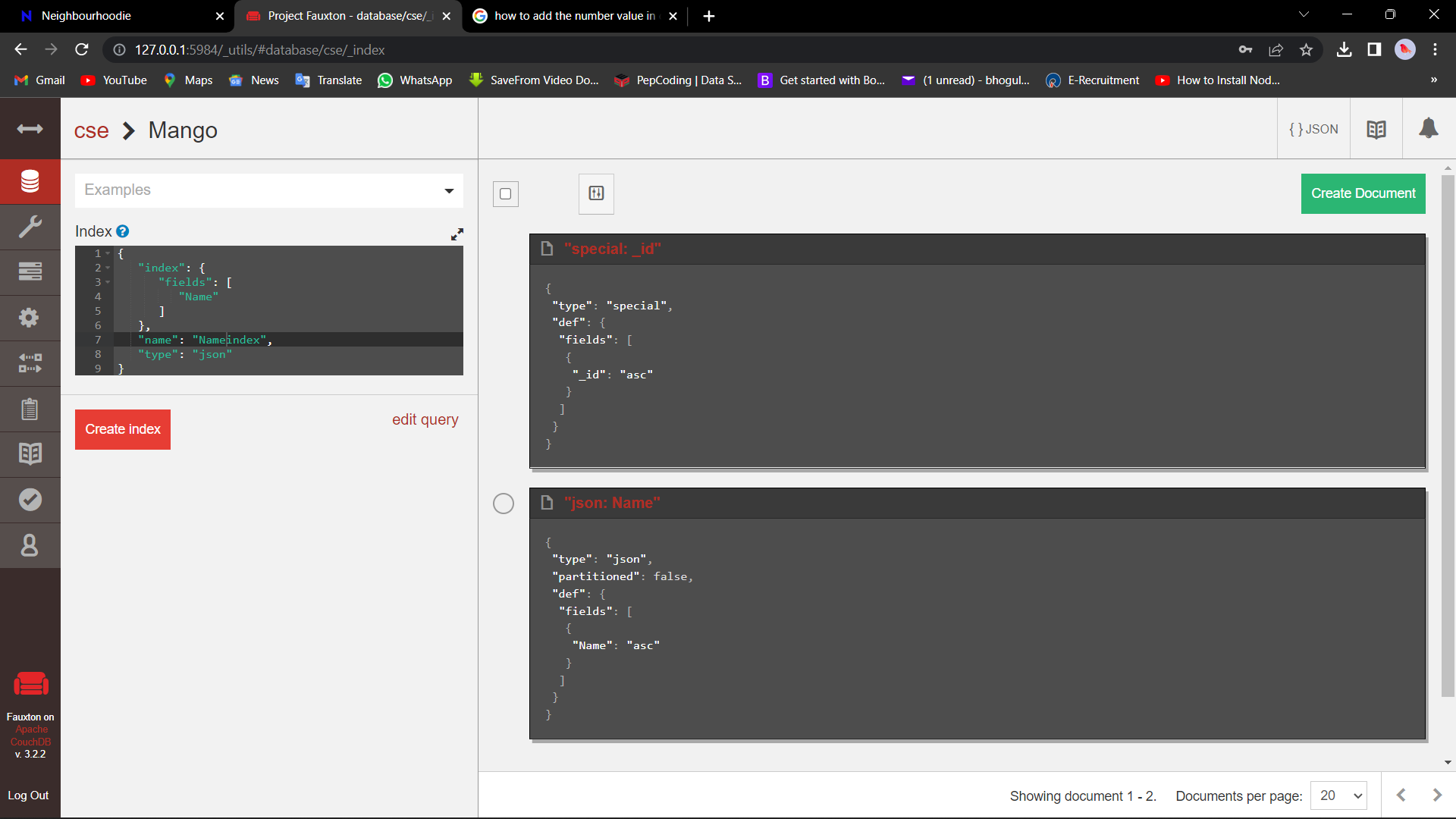
1. Display names and marks of students having marks equal to or greater than 80 and less than 90:



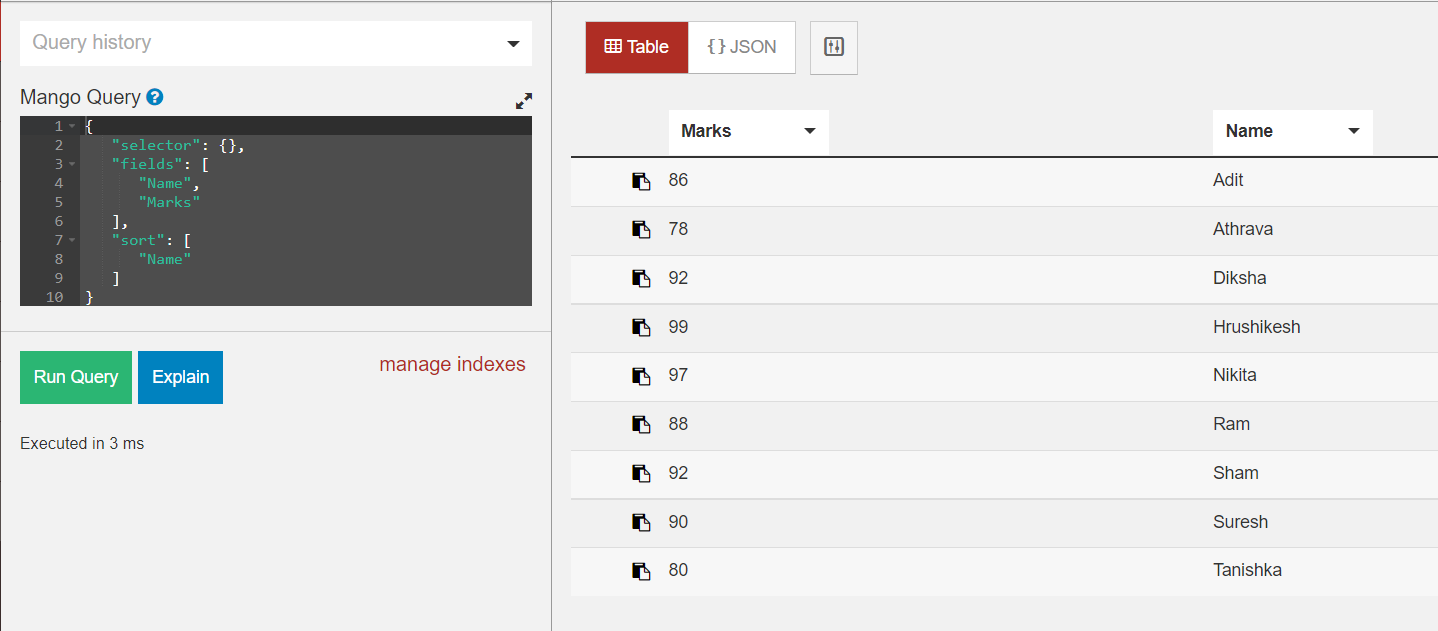
1. Display the name and mark of the student having Roll no 1:

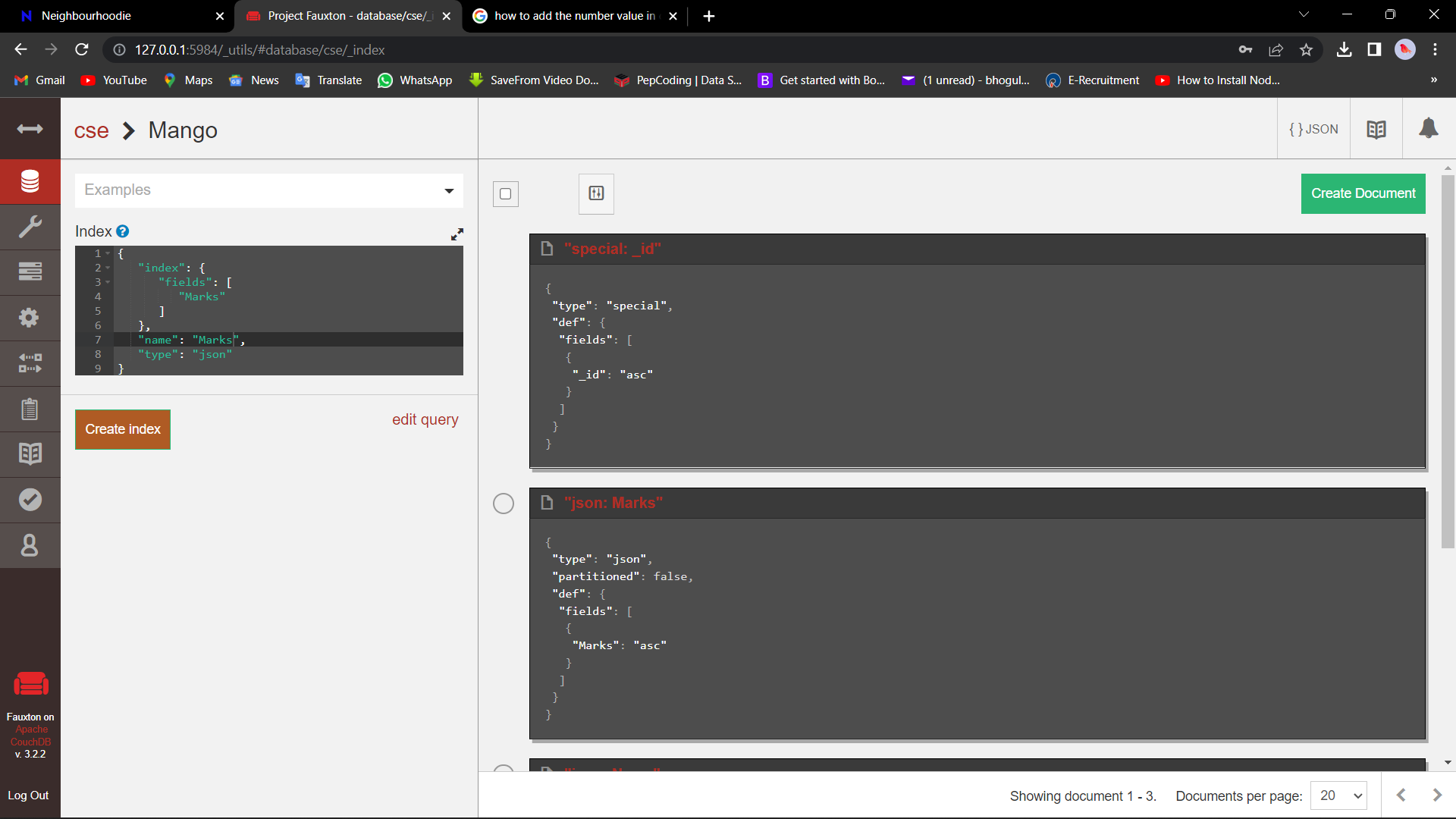


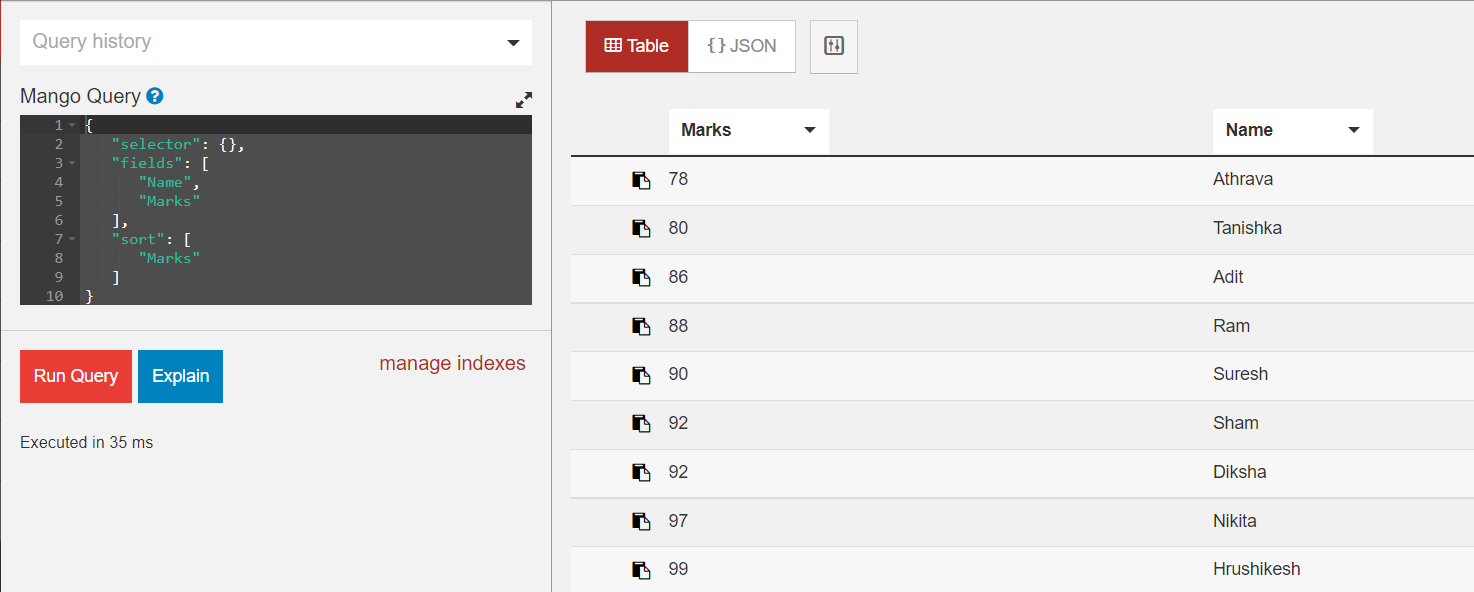
1. Create an index on the name:



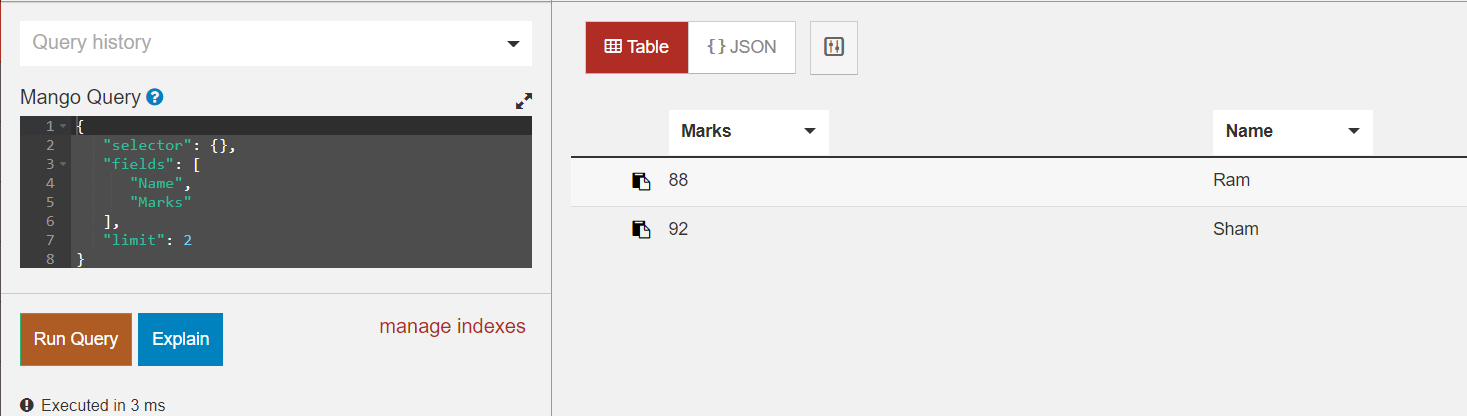
1. Display documents in ascending order of name:



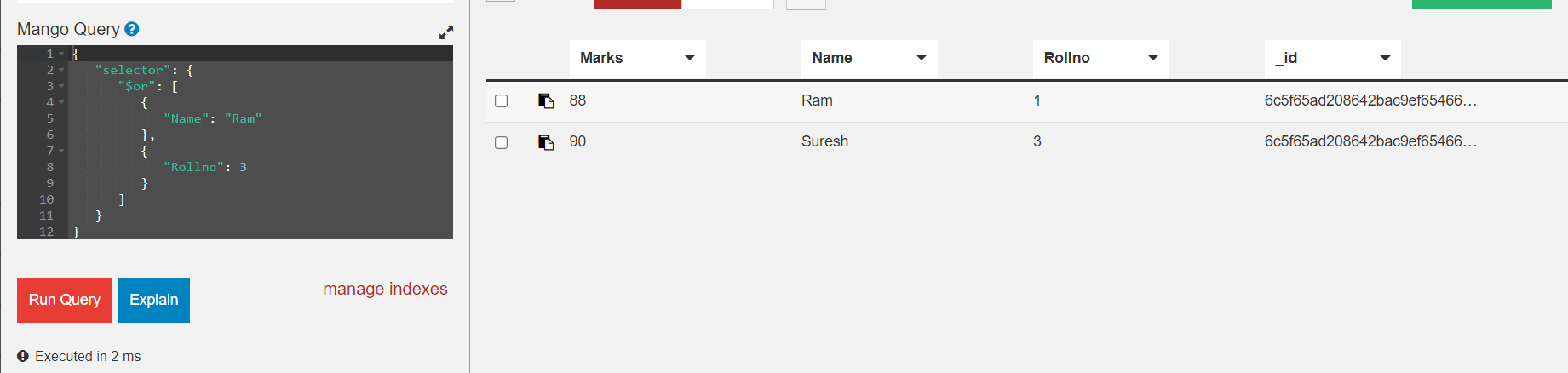
1. Create an index on marks: 
2. Display documents in ascending order of marks:



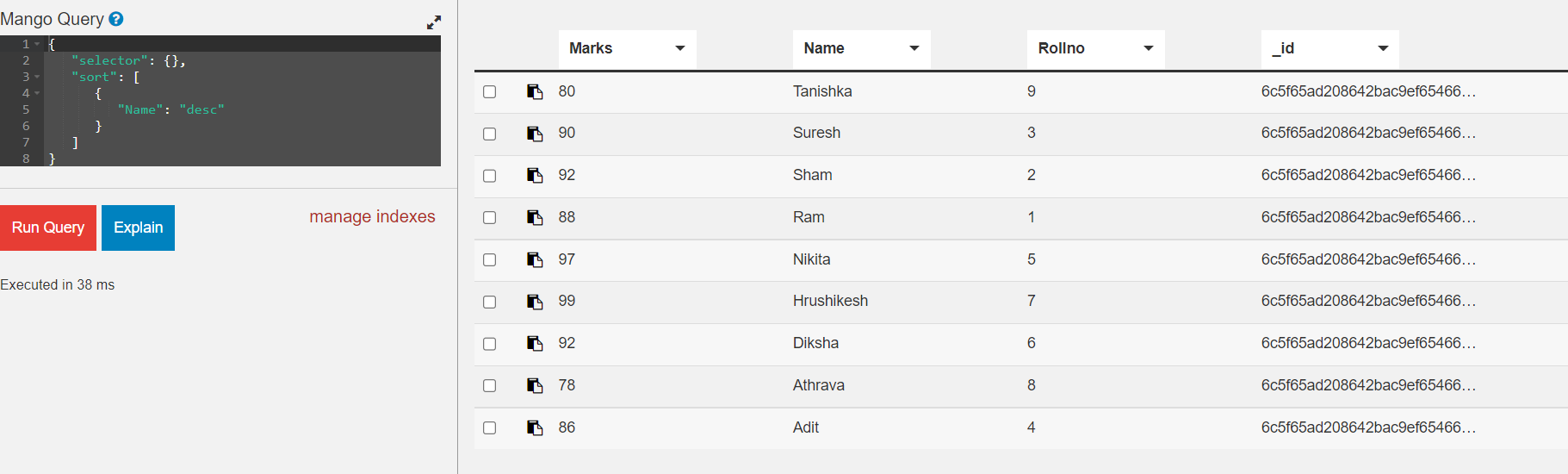
1. Display the first 2 documents:



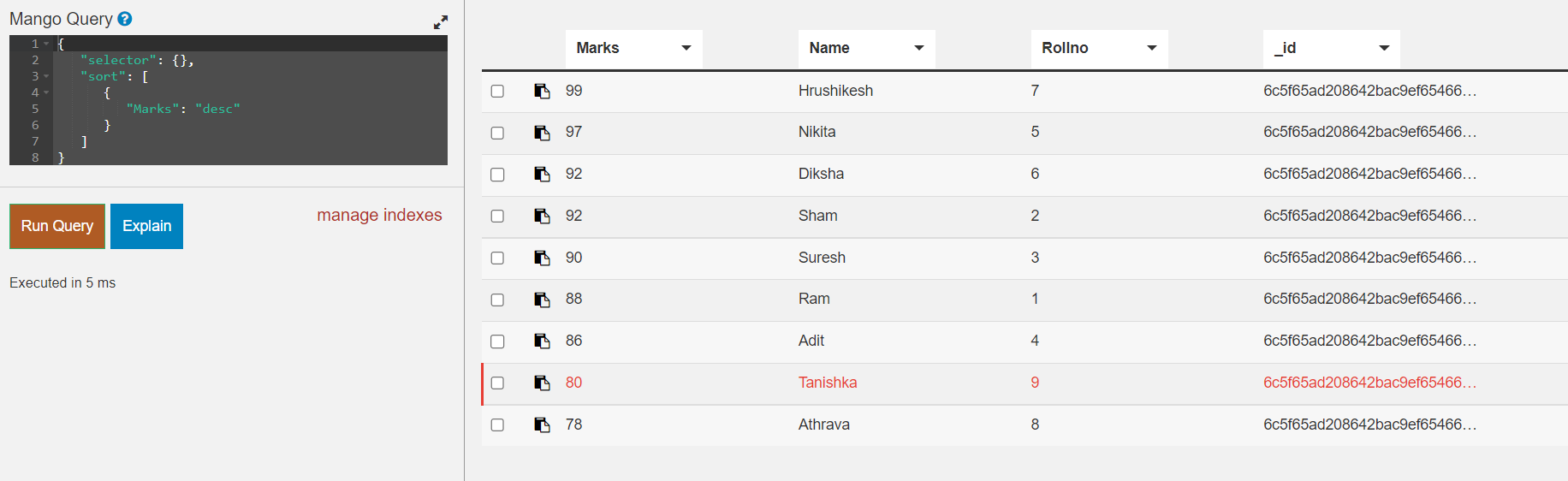
1. Display the document of Ram or Roll no 3:



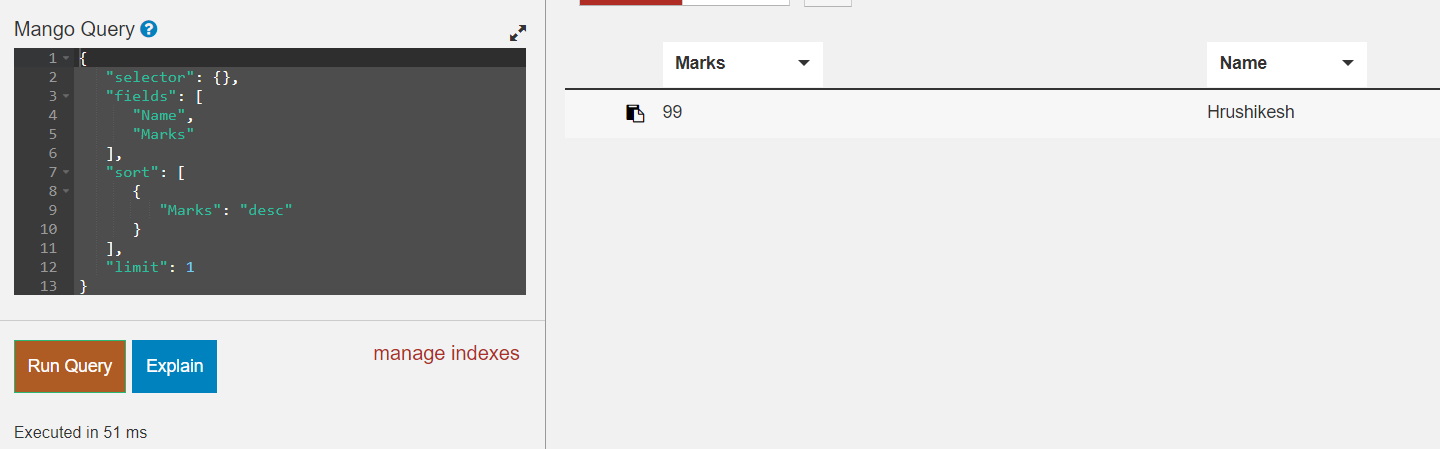
1. Display all the documents in descending order of name:



1. Display all the documents in descending order of the marks:



1. Display names and marks of students having the highest marks:



**Assignment 3: Part 1.2 - Semijoin operation program**

import java.sql.\*;

class bank

{

public static void main(String args[])

{

int acc\_no=0;

String cust\_name;

String branch;

float balance;

try

{

Class.forName("com.mysql.jdbc.Driver");

Connection con;

con=DriverManager.getConnection("jdbc:mysql://localhost:3306/DYP\_Bank?useSSL=false","root","root");

Statement stmt1=con.createStatement();

Statement stmt2=con.createStatement();

Statement stmt3=con.createStatement();

Statement stmt4=con.createStatement();

ResultSet rs=stmt1.executeQuery("select acc\_no from depositor");

while(rs.next())

{

acc\_no= rs.getInt(1);

stmt2.executeUpdate("insert into temp1 values("+acc\_no+");");

}

ResultSet rs1=stmt1.executeQuery("select \* from account natural join temp1");

while(rs1.next())

{

acc\_no= rs1.getInt(1);

branch=rs1.getString(2);

balance=rs1.getFloat(3);

stmt3.executeUpdate("insert into temp2 values("+acc\_no+",'"+branch+"',"+balance+");");

}

ResultSet rs2=stmt1.executeQuery("select \* from depositor natural join temp2");

while(rs2.next())

{

acc\_no= rs2.getInt(1);

cust\_name=rs2.getString(2);

branch=rs2.getString(3);

balance=rs2.getFloat(4);

stmt4.executeUpdate("insert into resultset values("+acc\_no+",'"+cust\_name+"','"+branch+"',"+balance+");");

}

System.out.println("Semijoin Result:");

ResultSet rs3 = stmt1.executeQuery("select \* from resultset");

while(rs3.next()){

acc\_no=rs3.getInt(1);

cust\_name=rs3.getString(2);

branch=rs3.getString(3);

balance=rs3.getFloat(4);

System.out.println(acc\_no+"\t"+cust\_name+"\t"+branch+"\t"+balance);

}

con.close();

}

catch(Exception e){ System.out.println(e);

}

}

}

**Output:**

Semijoin Result:

120 Aditi Rajarampuri 10000.0

122 Vicky Shahupuri 20000.2

123 Ram Tarabai Park 25000.3

**Experiment No : 01**

**MYSQL Installation on Ubuntu**

sudo apt update

sudo apt upgrade

sudo apt install mysql-server

mysql --version

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Set Password**

sudo mysql\_secure\_installation

password- root

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Enter in Mysql**

sudo mysql -u root

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Solve following Queries:**

**Consider the following schema :**

employee(empid, empname, salary, designation, company-name)

**Create database Company :**

mysql> create database company;

Query OK, 1 row affected (0.12 sec)

mysql> show databases;

+--------------------------+

| Database |

+--------------------------+

| Bank |

| Customers |

| Horizontal\_Fragmentation |

| company |

| employee |

| exp9 |

| information\_schema |

| mysql |

| performance\_schema |

| sys |

+--------------------------+

10 rows in set (0.06 sec)

**Execute the following queries:**

**1) Create employee table :**

mysql> use company;

Database changed

create table employee(empid int,empname varchar(20),salary float,designation varchar(20),company\_name varchar(20));

Query OK, 0 rows affected (0.93 sec)

**2) Add city column into employee :**

mysql> alter table employee add(city varchar(25));

Query OK, 0 rows affected (0.39 sec)

Records: 0 Duplicates: 0 Warnings: 0

**3) Insert at least 10 records with meaningful data.**

mysql> insert into employee values(101,"Sakshi",50000,"CEO","TCS","Pune");

Query OK, 1 row affected (0.22 sec)

mysql> insert into employee values(102,"Chirag",50000,"CEO","Tech Mahindra","Pune");

Query OK, 1 row affected (0.12 sec)

mysql> insert into employee values(103,"Shivani",40000,"Manager","KPIT","Mumbai");

Query OK, 1 row affected (0.12 sec)

mysql> insert into employee values(104,"Ashu",40000,"Manager","DXC","Kolhapur");

Query OK, 1 row affected (0.12 sec)

mysql> insert into employee values(105,"Swara",20000,"HR","DXC","Solapur");

Query OK, 1 row affected (0.13 sec)

mysql> insert into employee values(106,"Swayam",21000,"HR","VOIS","Kolhapur");

Query OK, 1 row affected (0.15 sec)

mysql> insert into employee values(107,"Sara",9000,"Supervisor","Tech Mahindra","Pune");

Query OK, 1 row affected (0.10 sec)

mysql> insert into employee values(108,"Swati",30000,"employee","Cognizant","Mumbai");

Query OK, 1 row affected (0.15 sec)

mysql> insert into employee values(109,"Ram",30000,"employee","DXC","Mumbai");

Query OK, 1 row affected (0.10 sec)

mysql> insert into employee values(110,"Piyush",14000,"employee11","TCS","Pune");

Query OK, 1 row affected (0.14 sec)

mysql> select \* from employee;

+-------+----------+--------+-------------+---------------+----------+

| empid | empname | salary | designation | company\_name | city |

+-------+----------+--------+-------------+---------------+----------+

| 101 | Sakshi | 50000 | CEO | TCS | Pune |

| 102 | Chirag | 50000 | CEO | Tech Mahindra | Pune |

| 103 | Shivani | 40000 | Manager | KPIT | Mumbai |

| 104 | Ashu | 40000 | Manager | DXC | Kolhapur |

| 105 | Swara | 20000 | HR | DXC | Solapur |

| 106 | Swayam | 21000 | HR | VOIS | Kolhapur |

| 107 | Sara | 9000 | Supervisor | Tech Mahindra | Pune |

| 108 | Swati | 30000 | employee | Cognizant | Mumbai |

| 109 | Ram | 30000 | employee | DXC | Mumbai |

| 110 | Piyush | 14000 | employee11 | TCS | Pune |

+-------+----------+--------+-------------+---------------+----------+

10 rows in set (0.00 sec)

**4) Display all employees with their salary :**

mysql> select empname,salary from employee;

+----------+--------+

| empname | salary |

+----------+--------+

| Sakshi | 50000 |

| Chirag | 50000 |

| Shivani | 40000 |

| Ashu | 40000 |

| Swara | 20000 |

| Swayam | 21000 |

| Sara | 9000 |

| Swati | 30000 |

| Ram | 30000 |

| Piyush | 14000 |

+----------+--------+

10 rows in set (0.00 sec)

**5) Find the name of the employee along with their id :**

mysql> select empid,empname from employee;

+-------+----------+

| empid | empname |

+-------+----------+

| 101 | Sakshi |

| 102 | Chirag |

| 103 | Shivani |

| 104 | Ashu |

| 105 | Swara |

| 106 | Swayam |

| 107 | Sara |

| 108 | Swati |

| 109 | Ram |

| 110 | Piyush |

+-------+----------+

10 rows in set (0.00 sec)

**6) Find employees who are living in “Kolhapur” :**

mysql> select \* from employee where city="Kolhapur";

+-------+----------+--------+-------------+--------------+----------+

| empid | empname | salary | designation | company\_name | city |

+-------+----------+--------+-------------+--------------+----------+

| 104 | Ashu | 40000 | Manager | DXC | Kolhapur |

| 106 | Swayam | 21000 | HR | VOIS | Kolhapur |

+-------+----------+--------+-------------+--------------+----------+

2 rows in set (0.00 sec)

**7) Find name of the employees whose salary is >10000 :**

mysql> select \* from employee where salary>10000;

+-------+----------+--------+-------------+---------------+----------+

| empid | empname | salary | designation | company\_name | city |

+-------+----------+--------+-------------+---------------+----------+

| 101 | Sakshi | 50000 | CEO | TCS | Pune |

| 102 | Chirag | 50000 | CEO | Tech Mahindra | Pune |

| 103 | Shivani | 40000 | Manager | KPIT | Mumbai |

| 104 | Ashu | 40000 | Manager | DXC | Kolhapur |

| 105 | Swara | 20000 | HR | DXC | Solapur |

| 106 | Swayam | 21000 | HR | VOIS | Kolhapur |

| 108 | Swati | 30000 | employee | Cognizant | Mumbai |

| 109 | Ram | 30000 | employee | DXC | Mumbai |

| 110 | Piyush | 14000 | employee11 | TCS | Pune |

+-------+----------+--------+-------------+---------------+----------+

9 rows in set (0.01 sec)

**8) Find name of the employees whose salary is <15000 :**

mysql> select \* from employee where salary<15000;

+-------+----------+--------+-------------+---------------+------+

| empid | empname | salary | designation | company\_name | city |

+-------+----------+--------+-------------+---------------+------+

| 107 | Sara | 9000 | Supervisor | Tech Mahindra | Pune |

| 110 | Piyush | 14000 | employee11 | TCS | Pune |

+-------+----------+--------+-------------+---------------+------+

2 rows in set (0.00 sec)

**9) Update designation of employee 11 to ‘Asst. Prof.’ :**

mysql> update employee set designation="AsstProf" where designation="employee11";

Query OK, 1 row affected (0.11 sec)

Rows matched: 1 Changed: 1 Warnings: 0

mysql> select \* from employee where designation="AsstProf";

+-------+----------+--------+-------------+--------------+------+

| empid | empname | salary | designation | company\_name | city |

+-------+----------+--------+-------------+--------------+------+

| 110 | Piyush | 14000 | AsstProf | TCS | Pune |

+-------+----------+--------+-------------+--------------+------+

1 row in set (0.00 sec)

**10) Delete the employees having designation “supervisor” :**

mysql> delete from employee where designation="Supervisor";

Query OK, 1 row affected (0.22 sec)

mysql> select \* from employee where designation="Supervisor";

Empty set (0.00 sec)

**11) Increment the salary of employees by 5% :**

mysql> update employee set salary=salary+(salary\*5/100);

Query OK, 9 rows affected (0.17 sec)

Rows matched: 9 Changed: 9 Warnings: 0

mysql> select \* from employee;

+-------+----------+--------+-------------+---------------+----------+

| empid | empname | salary | designation | company\_name | city |

+-------+----------+--------+-------------+---------------+----------+

| 101 | Sakshi | 52500 | CEO | TCS | Pune |

| 102 | Chirag | 52500 | CEO | Tech Mahindra | Pune |

| 103 | Shivani | 42000 | Manager | KPIT | Mumbai |

| 104 | Ashu | 42000 | Manager | DXC | Kolhapur |

| 105 | Swara | 21000 | HR | DXC | Solapur |

| 106 | Swayam | 22050 | HR | VOIS | Kolhapur |

| 108 | Swati | 31500 | employee | Cognizant | Mumbai |

| 109 | Ram | 31500 | employee | DXC | Mumbai |

| 110 | Piyush | 14700 | AsstProf | TCS | Pune |

+-------+----------+--------+-------------+---------------+----------+

9 rows in set (0.00 sec)

**12) Find the name of employees having salary between 18000 and 22000 :**

mysql> select empname from employee where salary between 18000 and 22000;

+----------+

| empname |

+----------+

| Swara |

| Swati |

+----------+

2 rows in set (0.00 sec)

**13) Find number of employees working in each company :**

mysql> select company\_name, count(\*) from employee group by company\_name;

+---------------+----------+

| company\_name | count(\*) |

+---------------+----------+

| TCS | 2 |

| Tech Mahindra | 1 |

| KPIT | 1 |

| DXC | 3 |

| VOIS | 1 |

| Cognizant | 1 |

+---------------+----------+

6 rows in set (0.05 sec)

**14) Find employees having same salary as Ram :**

select \* from employee where salary IN(select salary from employee group by salary having count(salary)>1) and city="Mumbai";

+-------+----------+--------+-------------+--------------+--------+

| empid | empname | salary | designation | company\_name | city |

+-------+----------+--------+-------------+--------------+--------+

| 108 | Swati | 31500 | employee | Cognizant | Mumbai |

| 109 | Ram | 31500 | employee | DXC | Mumbai |

+-------+----------+--------+-------------+--------------+--------+

2 rows in set (0.00 sec)

**15) Find employee having maximum salary :**

mysql> select \* from employee where salary in (select max(salary)from employee);

+-------+----------+--------+-------------+---------------+------+

| empid | empname | salary | designation | company\_name | city |

+-------+----------+--------+-------------+---------------+------+

| 101 | Sakshi | 52500 | CEO | TCS | Pune |

| 102 | Chirag | 52500 | CEO | Tech Mahindra | Pune |

+-------+----------+--------+-------------+---------------+------+

2 rows in set (0.00 sec)

**16) Find company having highest average salary :**

mysql> select company\_name from employee where salary in(select max(salary)from employee);

+---------------+

| company\_name |

+---------------+

| TCS |

| Tech Mahindra |

+---------------+

2 rows in set (0.00 sec)

mysql> select company\_name from employee where salary>(select avg(salary)from

employee);

+---------------+

| company\_name |

+---------------+

| TCS |

| Tech Mahindra |

| KPIT |

| DXC |

+---------------+

4 rows in set (0.00 sec)

**Experiment No : 05**

**1. Even or Odd:**

SQL> connect system/manager;

Connected.

SQL> set serveroutput on;

SQL> declare

2 num1 number := &num1;

3 begin

4 if(mod(num1,2)=0)

5 then

6 dbms\_output.put\_line('num1 is even'||num1);

7 else

8 dbms\_output.put\_line('num1 is odd'||num1);

9 end if;

10 end;

11 /

Enter value for num1: 4

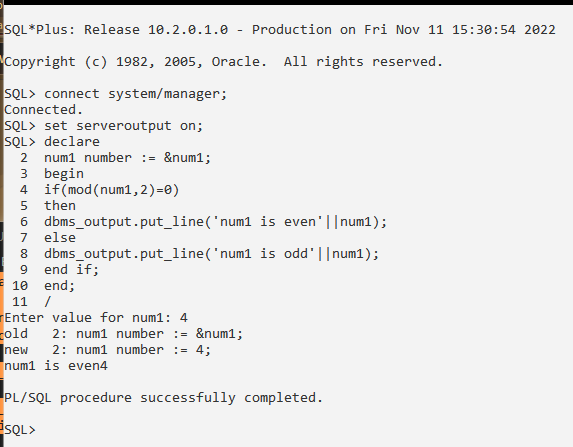
old 2: num1 number := &num1;

new 2: num1 number := 4;

num1 is even4

PL/SQL procedure successfully completed.

SQL>



**2.Stored Procedure :**

SQL\*Plus: Release 10.2.0.1.0 - Production on Fri Nov 11 15:30:54 2022

Copyright (c) 1982, 2005, Oracle. All rights reserved.

SQL> connect system/manager;

Connected.

SQL> set serveroutput on;

SQL> declare

2 num1 number := &num1;

3 begin

4 if(mod(num1,2)=0)

5 then

6 dbms\_output.put\_line('num1 is even'||num1);

7 else

8 dbms\_output.put\_line('num1 is odd'||num1);

9 end if;

10 end;

11 /

Enter value for num1: 4

old 2: num1 number := &num1;

new 2: num1 number := 4;

num1 is even4

PL/SQL procedure successfully completed.

SQL> create or replace procedure factorial2

2 (num1 in number)

3 is

4 fact1 number:=1;

5 begin

6 for i in 1..num1

7 loop

8 fact1:=fact1\*i;

9 end loop;

10 dbms\_output.put\_line('fact is:'||fact1);

11 end factorial2;

12 /

Procedure created.

SQL> declare

2 num1 number:=5;

3 begin

4 factorial2(num1);

5 end;

6 /

fact is:120

PL/SQL procedure successfully completed.

**3.Stored Function :**

SQL> create or replace function factorial

2 (num1 in number)

3 return number

4 is

5 fact1 number := 1;

6 begin

7 for i in 1..num1

8 loop

9 fact1 := fact1\*i;

10 end loop;

11 return fact1;

12 end factorial;

13 /

Function created.

SQL> declare

2 num1 number:= 6;

3 f number(5);

4 begin

5 f:= factorial(num1);

6 dbms\_output.put\_line('The factorial is :'||f);

7 end;

8 /

The factorial is :720

PL/SQL procedure successfully completed.

**4.Cursor :**

**Q. Write a PL/SQL cursor if salary of employee is greater than 10000 then increment it by 6 % otherwise by 5%.**

**Query:**

SQL\*Plus: Release 10.2.0.1.0 - Production on Tue Nov 15 11:52:02 2022

Copyright (c) 1982, 2005, Oracle. All rights reserved.

SQL> connect system/manager;

Connected.

SQL> set serveroutput on;

SQL> create table emp(eid int,ename varchar2(20),esalary number(5));

Table created.

SQL> insert into emp values(11,'Sakshi',30000);

1 row created.

SQL> insert into emp values(12,'Chirag',50000);

1 row created.

SQL> insert into emp values(13,'Rutika',20000);

1 row created.

SQL> insert into emp values(14,'Pallavi',9000);

1 row created.

SQL> insert into emp values(15,'Anamika',8000);

1 row created.

SQL> select \* from emp;

EID ENAME ESALARY

---------- -------------------- ----------

11 Sakshi 30000

12 Chirag 50000

13 Rutika 20000

14 Pallavi 9000

15 Anamika 8000

SQL> declare

2 empid emp.eid %type;

3 esal emp.esalary %type;

4 cursor s1 is select eid,esalary from emp;

5 begin

6 open s1;

7 loop

8 fetch s1 into empid,esal;

9 if(esal > 10000)

10 then

11 esal:= esal \* 1.06;

12 else

13 esal:= esal \* 1.05;

14 end if;

15 update emp set esalary=esal where eid=empid;

16 exit when s1 %notfound;

17 end loop;

18 close s1;

19 end;

20 /

PL/SQL procedure successfully completed.

SQL> select \* from emp;

EID ENAME ESALARY

---------- -------------------- ----------

11 Sakshi 31800

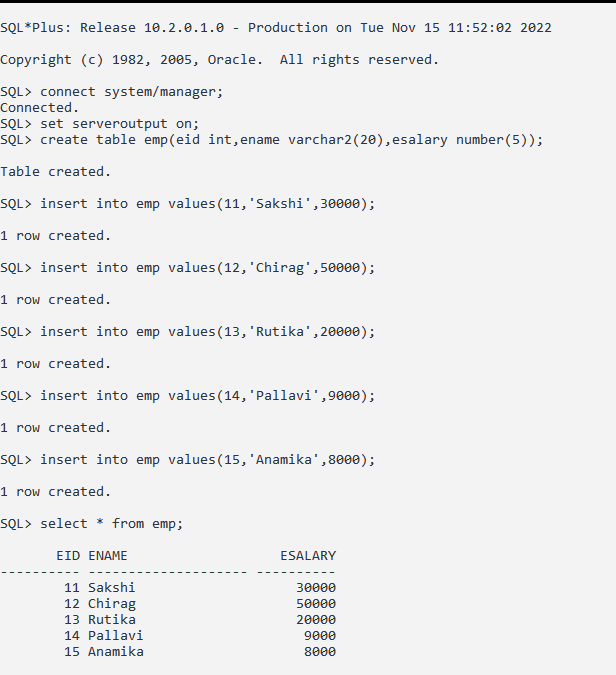
12 Chirag 53000

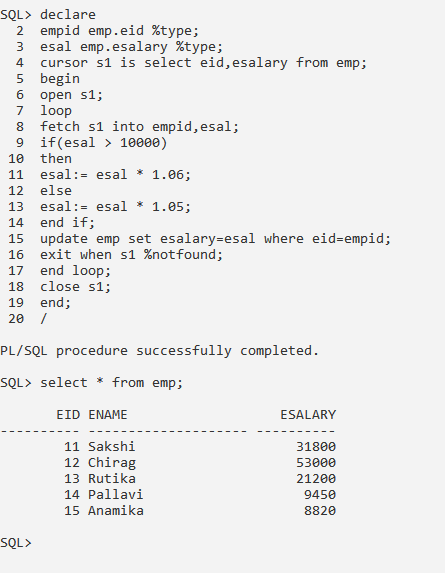
13 Rutika 21200

14 Pallavi 9450

15 Anamika 8820

SQL>





**Q. Write a cursor to calculate net Salary for above emp table.**

SQL> alter table emp add netsalary number(5);

Table altered.

SQL> alter table emp modify netsalary numeric(10,2);

Table altered.

SQL> declare

2 empid emp.eid %type;

3 esal emp.esalary %type;

4 nsal emp.netsalary %type;

5 cursor s1 is select eid,esalary from emp;

6 begin

7 open s1;

8 loop

9 fetch s1 into empid,esal;

10 nsal:=esal+(esal\*0.15)+(esal\*0.60)+(esal\*0.2);

11 update emp set netsalary = nsal where eid=empid;

12 exit when s1 %notfound;

13 end loop;

14 close s1;

15 end;

16 /

PL/SQL procedure successfully completed.

SQL> select \* from emp;

EID ENAME ESALARY NETSALARY

---------- -------------------- ---------- ----------

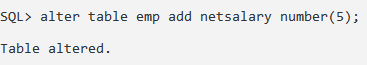
11 Sakshi 31800 62010

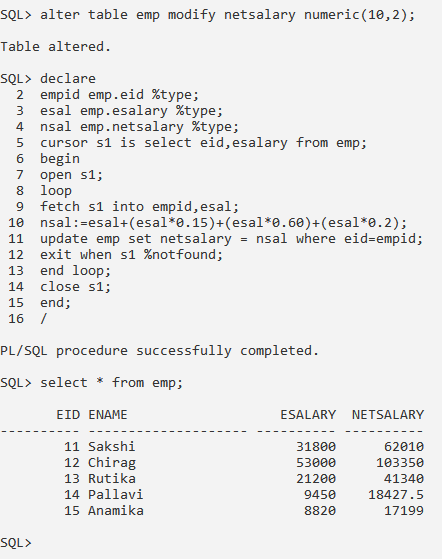
12 Chirag 53000 103350

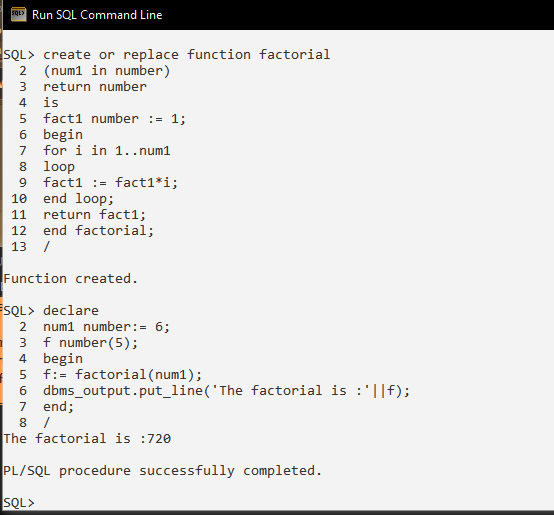
13 Rutika 21200 41340

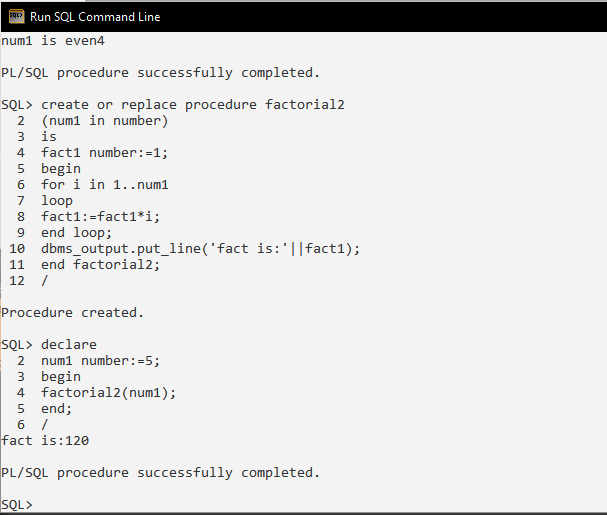
14 Pallavi 9450 18427.5

15 Anamika 8820 17199









**5.Trigger :**

**Trigger :**

**Example: Trigger before the insertion of the student record in the table:**

SQL\*Plus: Release 10.2.0.1.0 - Production on Tue Nov 15 12:45:38 2022

Copyright (c) 1982, 2005, Oracle. All rights reserved.

SQL> connect system/manager;

Connected.

SQL> set serveroutput on;

SQL> CREATE TABLE Student

2 (

3 RollNo INT NOT NULL PRIMARY KEY,

4 FirstName Varchar (20),

5 EnglishMarks number(5),

6 PhysicsMarks number(5),

7 ChemistryMarks number(5),

8 MathsMarks number(5),

9 TotalMarks number(5),

10 Percentage numeric(4,2)

11 );

Table created.

SQL>

SQL> CREATE or replace TRIGGER Student\_Marks1

2 BEFORE INSERT ON Student

3 FOR EACH ROW

4 begin

5 :new.TotalMarks := :new.EnglishMarks + :new.PhysicsMarks + :new.ChemistryMarks + :new.MathsMarks;

6 :new.Percentage := (:new.TotalMarks / 400) \* 100;

7 end;

8 /

Trigger created.

SQL> INSERT INTO Student (RollNo, FirstName, EnglishMarks, PhysicsMarks, ChemistryMarks, MathsMarks, TotalMarks, Percentage) VALUES ( 64, 'Sakshi', 88, 75, 69, 92, 0, 0);

1 row created.

SQL> INSERT INTO Student (RollNo, FirstName, EnglishMarks, PhysicsMarks, ChemistryMarks, MathsMarks, TotalMarks, Percentage) VALUES ( 78, 'Rutika', 88, 75, 60, 90, 0, 0);

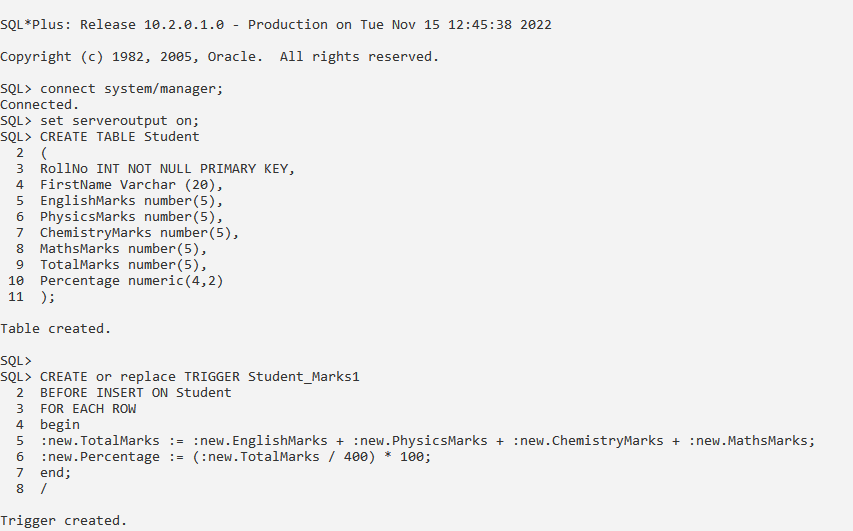
1 row created.

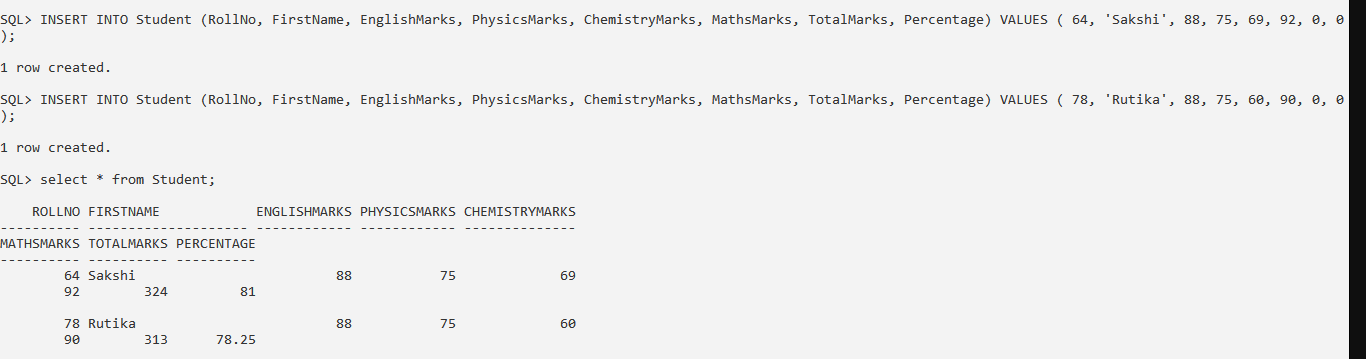
SQL> select \* from Student;

ROLLNO FIRSTNAME ENGLISHMARKS PHYSICSMARKS CHEMISTRYMARKS MATHSMARKS TOTALMARKS PERCENTAGE

64 Sakshi 88 75 69 92 324 81

78 Rutika 88 75 60 90 313 78.25





**Example 2 :** **Check the salary difference by procedure:**

SQL> create table customer (id number(5), name varchar(15), age number(5), address varchar(15), salary numeric(8,2));

Table created.

SQL> insert into customer values (1, 'Sakshi', 21, 'Kolhapur', 20000);

1 row created.

SQL> insert into customer values (2, 'Rutika', 22, 'Pune', 30000);

1 row created.

SQL> insert into customer values (3, 'Chirag', 23, 'Kolhapur', 25000);

1 row created.

SQL> select \* from customer;

ID NAME AGE ADDRESS SALARY

---------- --------------- ---------- --------------- ----------

1 Sakshi 21 Kolhapur 20000

2 Rutika 22 Pune 30000

3 Chirag 23 Kolhapur 25000

SQL> CREATE OR REPLACE TRIGGER display\_salary\_changes

2 BEFORE DELETE OR INSERT OR UPDATE ON customer

3 FOR EACH ROW

4 WHEN (NEW.id > 0)

5 DECLARE

6 sal\_diff number;

7 BEGIN

8 sal\_diff := :NEW.salary - :OLD.salary;

9 dbms\_output.put\_line('Old salary: ' || :OLD.salary);

10 dbms\_output.put\_line('New salary: ' || :NEW.salary);

11 dbms\_output.put\_line('Salary difference: ' || sal\_diff);

12 END;

13 /

Trigger created.

SQL> DECLARE

2 total\_rows number(2);

3 BEGIN

4 UPDATE customer

5 SET salary = salary + 5000;

6 IF sql%notfound THEN

7 dbms\_output.put\_line('no customers updated');

8 ELSIF sql%found THEN

9 total\_rows := sql%rowcount;

10 dbms\_output.put\_line( total\_rows || ' customers updated ');

11 END IF;

12 END;

13 /

Old salary: 20000

New salary: 25000

Salary difference: 5000

Old salary: 30000

New salary: 35000

Salary difference: 5000

Old salary: 25000

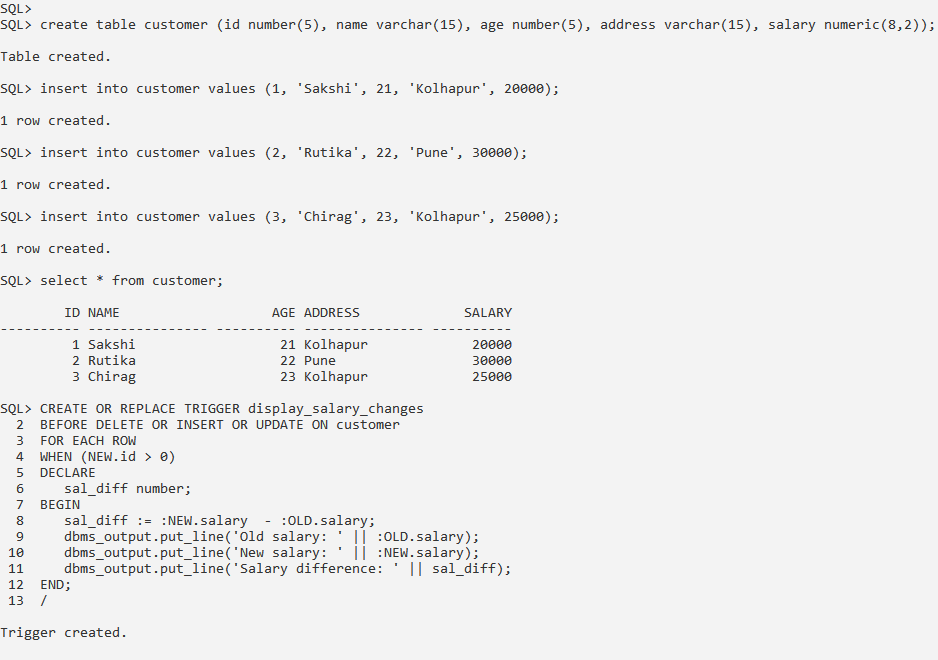
New salary: 30000

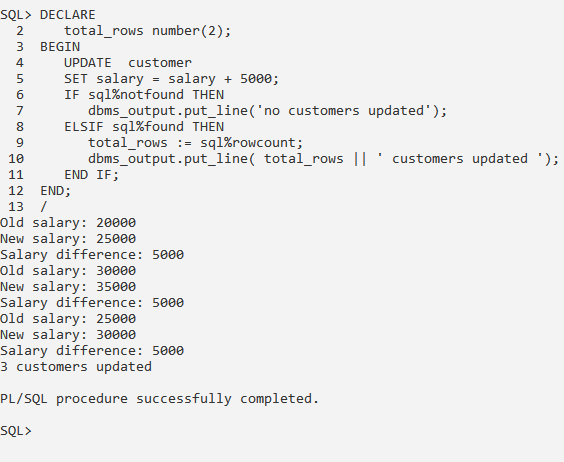
Salary difference: 5000

3 customers updated

PL/SQL procedure successfully completed.

SQL>





Assignment no. 7

MongoDB Queries

**Create db Employee**

> use cse switched to db cse

> db.createCollection("Employee")

{ "ok" : 1 }

> show collections

Employee

**Insert 10 employee documents**

>

db.Employee.insert({Firstname:"Ashutosh",Lastname:"Zende",gender:"F",salary:30000,deptname:" CSE"})

WriteResult({ "nInserted" : 1 })

>

db.Employee.insert({Firstname:"Anay",Lastname:"Zende",gender:"M",salary:25000,deptname:" CSE"})

WriteResult({ "nInserted" : 1 })

>

db.Employee.insert({Firstname:"Smita",Lastname:"Patil",gender:"F",salary:25000,deptname:" Mechanical"})

WriteResult({ "nInserted" : 1 })

>

db.Employee.insert({Firstname:"Tejas",Lastname:"Patil",gender:"M",salary:30000,deptname:"Rob otics"})

WriteResult({ "nInserted" : 1 })

>

db.Employee.insert({Firstname:"Abhishek",Lastname:"Mane",gender:"M",salary:35000,deptnam e:"ENTC"})

WriteResult({ "nInserted" : 1 })

>

db.Employee.insert({Firstname:"Ashutosh",Lastname:"Patil",gender:"M",salary:15000,deptname:" CSE"})

WriteResult({ "nInserted" : 1 })

>

db.Employee.insert({Firstname:"Sakshi",Lastname:"Bhosale",gender:"F",salary:35000,deptname:" CSE"})

WriteResult({ "nInserted" : 1 })

>

db.Employee.insert({Firstname:"Mahesh",Lastname:"Bhosale",gender:"M",salary:45000,deptname:" Mech"})

WriteResult({ "nInserted" : 1 })

>

db.Employee.insert({Firstname:" Prathmesh",Lastname:"Shinde",gender:"F",salary:45000,deptname:"M ech"})

WriteResult({ "nInserted" : 1 })>

db.Employee.insert({Firstname:"Girish",Lastname:"Gaikwad",gender:"F",salary:35000,deptnam e:"CSE"})

WriteResult({ "nInserted" : 1 })

> db.Employee.find()

{ "\_id" : ObjectId("638dde0d5b9cae48f3c90016"), "Firstname" : "Ashutosh", "Lastname" :

"Zende", "gender" : "F", "salary" : 30000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638dde535b9cae48f3c90017"), "Firstname" : "Anay", "Lastname" : "Zende",

"gender" : "M", "salary" : 25000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638ddef25b9cae48f3c9001a"), "Firstname" : "Smita", "Lastname" : "Patil",

"gender" : "F", "salary" : 25000, "deptname" : "Mechanical" }

{ "\_id" : ObjectId("638ddf405b9cae48f3c9001b"), "Firstname" : "Tejas", "Lastname" : "Patil",

"gender" : "M", "salary" : 30000, "deptname" : "Robotics" }

{ "\_id" : ObjectId("638de1655b9cae48f3c9001c"), "Firstname" : "Abhishek", "Lastname" :

"Mane", "gender" : "M", "salary" : 35000, "deptname" : "ENTC" }

{ "\_id" : ObjectId("638de1c05b9cae48f3c9001d"), "Firstname" : "Ashutosh", "Lastname" : "Patil",

"gender" : "M", "salary" : 15000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de1e75b9cae48f3c9001e"), "Firstname" : "Sakshi", "Lastname" : "Bhosale",

"gender" : "F", "salary" : 35000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de2245b9cae48f3c9001f"), "Firstname" : "Mahesh", "Lastname" : "Bhosale",

"gender" : "M", "salary" : 45000, "deptname" : "Mech" }

{ "\_id" : ObjectId("638de2685b9cae48f3c90021"), "Firstname" : " Prathmesh", "Lastname" : "Shinde",

"gender" : "F", "salary" : 45000, "deptname" : "Mech" }

{ "\_id" : ObjectId("638de29e5b9cae48f3c90022"), "Firstname" : "Girish", "Lastname" :

"Gaikwad", "gender" : "F", "salary" : 35000, "deptname" : "CSE" }

**Salary greater than 20000**

> db.Employee.find({salary:{$gt:20000}});

{ "\_id" : ObjectId("638dde0d5b9cae48f3c90016"), "Firstname" : "Ashutosh", "Lastname" :

"Zende", "gender" : "F", "salary" : 30000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638dde535b9cae48f3c90017"), "Firstname" : "Anay", "Lastname" : "Zende",

"gender" : "M", "salary" : 25000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638ddef25b9cae48f3c9001a"), "Firstname" : "Smita", "Lastname" : "Patil",

"gender" : "F", "salary" : 25000, "deptname" : "Mechanical" }

{ "\_id" : ObjectId("638ddf405b9cae48f3c9001b"), "Firstname" : "Tejas", "Lastname" : "Patil",

"gender" : "M", "salary" : 30000, "deptname" : "Robotics" }

{ "\_id" : ObjectId("638de1655b9cae48f3c9001c"), "Firstname" : "Abhishek", "Lastname" :

"Mane", "gender" : "M", "salary" : 35000, "deptname" : "ENTC" }

{ "\_id" : ObjectId("638de1e75b9cae48f3c9001e"), "Firstname" : "Sakshi", "Lastname" : "Bhosale",

"gender" : "F", "salary" : 35000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de2245b9cae48f3c9001f"), "Firstname" : "Mahesh", "Lastname" : "Bhosale",

"gender" : "M", "salary" : 45000, "deptname" : "Mech" }

{ "\_id" : ObjectId("638de2685b9cae48f3c90021"), "Firstname" : " Prathmesh", "Lastname" : "Shinde",

"gender" : "F", "salary" : 45000, "deptname" : "Mech" }

{ "\_id" : ObjectId("638de29e5b9cae48f3c90022"), "Firstname" : "Girish", "Lastname" : "Gaikwad", "gender" : "F", "salary" : 35000, "deptname" : "CSE" }

**Salary less than 20000**

> db.Employee.find({salary:{$lt:20000}});

{ "\_id" : ObjectId("638de1c05b9cae48f3c9001d"), "Firstname" : "Ashutosh", "Lastname" : "Patil", "gender" : "M", "salary" : 15000, "deptname" : "CSE" }

**Employee of CSE DEPT**

> db.Employee.find({deptname:'CSE'});

{ "\_id" : ObjectId("638dde0d5b9cae48f3c90016"), "Firstname" : "Ashutosh", "Lastname" :

"Zende", "gender" : "F", "salary" : 30000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638dde535b9cae48f3c90017"), "Firstname" : "Anay", "Lastname" : "Zende", "gender" : "M", "salary" : 25000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de1c05b9cae48f3c9001d"), "Firstname" : "Ashutosh", "Lastname" : "Patil",

"gender" : "M", "salary" : 15000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de1e75b9cae48f3c9001e"), "Firstname" : "Sakshi", "Lastname" : "Bhosale",

"gender" : "F", "salary" : 35000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de29e5b9cae48f3c90022"), "Firstname" : "Girish", "Lastname" : "Gaikwad", "gender" : "F", "salary" : 35000, "deptname" : "CSE" }

**Salary of Prathmesh**

> db.Employee.find({Firstname:' Prathmesh'},{salary:1});

{ "\_id" : ObjectId("638de2685b9cae48f3c90021"), "salary" : 45000 }

**Max salary of dept**

> db.Employee.aggregate([{$group:{\_id:'$deptname',Maximumsalary:{$max:'$salary'}}}]);

{ "\_id" : "Mech", "Maximumsalary" : 45000 }

{ "\_id" : "CSE", "Maximumsalary" : 35000 }

{ "\_id" : "Robotics", "Maximumsalary" : 30000 }

{ "\_id" : "ENTC", "Maximumsalary" : 35000 }

{ "\_id" : "Mechanical", "Maximumsalary" : 25000 }

**Update Salary by 10%**

> db.Employee.update({},{$mul:{salary:1.1}},{multi:true});

WriteResult({ "nMatched" : 10, "nUpserted" : 0, "nModified" : 10 })

> db.Employee.find();

{ "\_id" : ObjectId("638dde0d5b9cae48f3c90016"), "Firstname" : "Ashutosh", "Lastname" :

"Zende", "gender" : "F", "salary" : 33000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638dde535b9cae48f3c90017"), "Firstname" : "Anay", "Lastname" : "Zende",

"gender" : "M", "salary" : 27500.000000000004, "deptname" : "CSE" }

{ "\_id" : ObjectId("638ddef25b9cae48f3c9001a"), "Firstname" : "Smita", "Lastname" : "Patil",

"gender" : "F", "salary" : 27500.000000000004, "deptname" : "Mechanical" }

{ "\_id" : ObjectId("638ddf405b9cae48f3c9001b"), "Firstname" : "Tejas", "Lastname" : "Patil",

"gender" : "M", "salary" : 33000, "deptname" : "Robotics" }

{ "\_id" : ObjectId("638de1655b9cae48f3c9001c"), "Firstname" : "Abhishek", "Lastname" :

"Mane", "gender" : "M", "salary" : 38500, "deptname" : "ENTC" }

{ "\_id" : ObjectId("638de1c05b9cae48f3c9001d"), "Firstname" : "Ashutosh", "Lastname" : "Patil",

"gender" : "M", "salary" : 16500, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de1e75b9cae48f3c9001e"), "Firstname" : "Sakshi", "Lastname" : "Bhosale",

"gender" : "F", "salary" : 38500, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de2245b9cae48f3c9001f"), "Firstname" : "Mahesh", "Lastname" : "Bhosale",

"gender" : "M", "salary" : 49500.00000000001, "deptname" : "Mech" }

{ "\_id" : ObjectId("638de2685b9cae48f3c90021"), "Firstname" : " Prathmesh", "Lastname" : "Shinde",

"gender" : "F", "salary" : 49500.00000000001, "deptname" : "Mech" }

{ "\_id" : ObjectId("638de29e5b9cae48f3c90022"), "Firstname" : "Girish", "Lastname" : "Gaikwad", "gender" : "F", "salary" : 38500, "deptname" : "CSE" }

**Update deptname of Prathmesh to CSE**

> db.Employee.update({Firstname:" Prathmesh"},{$set:{deptname:'CSE'}});

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.Employee.find();

{ "\_id" : ObjectId("638dde0d5b9cae48f3c90016"), "Firstname" : "Ashutosh", "Lastname" :

"Zende", "gender" : "F", "salary" : 33000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638dde535b9cae48f3c90017"), "Firstname" : "Anay", "Lastname" : "Zende",

"gender" : "M", "salary" : 27500.000000000004, "deptname" : "CSE" }

{ "\_id" : ObjectId("638ddef25b9cae48f3c9001a"), "Firstname" : "Smita", "Lastname" : "Patil",

"gender" : "F", "salary" : 27500.000000000004, "deptname" : "Mechanical" }

{ "\_id" : ObjectId("638ddf405b9cae48f3c9001b"), "Firstname" : "Tejas", "Lastname" : "Patil",

"gender" : "M", "salary" : 33000, "deptname" : "Robotics" }

{ "\_id" : ObjectId("638de1655b9cae48f3c9001c"), "Firstname" : "Abhishek", "Lastname" :

"Mane", "gender" : "M", "salary" : 38500, "deptname" : "ENTC" }

{ "\_id" : ObjectId("638de1c05b9cae48f3c9001d"), "Firstname" : "Ashutosh", "Lastname" : "Patil",

"gender" : "M", "salary" : 16500, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de1e75b9cae48f3c9001e"), "Firstname" : "Sakshi", "Lastname" : "Bhosale",

"gender" : "F", "salary" : 38500, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de2245b9cae48f3c9001f"), "Firstname" : "Mahesh", "Lastname" : "Bhosale",

"gender" : "M", "salary" : 49500.00000000001, "deptname" : "Mech" }

{ "\_id" : ObjectId("638de2685b9cae48f3c90021"), "Firstname" : " Prathmesh", "Lastname" : "Shinde",

"gender" : "F", "salary" : 49500.00000000001, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de29e5b9cae48f3c90022"), "Firstname" : "Girish", "Lastname" : "Gaikwad", "gender" : "F", "salary" : 38500, "deptname" : "CSE" }

**Display employees salary in descending order**

> db.Employee.find({},{Firstname:1,salary:1,\_id:0}).sort({salary:-1});

{ "Firstname" : "Mahesh", "salary" : 49500.00000000001 }

{ "Firstname" : " Prathmesh", "salary" : 49500.00000000001 }

{ "Firstname" : "Abhishek", "salary" : 38500 }

{ "Firstname" : "Sakshi", "salary" : 38500 }

{ "Firstname" : "Girish", "salary" : 38500 }

{ "Firstname" : "Ashutosh", "salary" : 33000 }

{ "Firstname" : "Tejas", "salary" : 33000 }

{ "Firstname" : "Anay", "salary" : 27500.000000000004 }

{ "Firstname" : "Smita", "salary" : 27500.000000000004 }

{ "Firstname" : "Ashutosh", "salary" : 16500 }

**Find employee whose salary is 33000**

> db.Employee.find({salary:{$eq:33000}});

{ "\_id" : ObjectId("638dde0d5b9cae48f3c90016"), "Firstname" : "Ashutosh", "Lastname" :

"Zende", "gender" : "F", "salary" : 33000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638ddf405b9cae48f3c9001b"), "Firstname" : "Tejas", "Lastname" : "Patil", "gender" : "M", "salary" : 33000, "deptname" : "Robotics" }

**Find top 5 employees having highest salary**

> db.Employee.find({},{Firstname:1,salary:1,\_id:0}).sort({salary:-1}).limit(5);

{ "Firstname" : " Prathmesh", "salary" : 49500.00000000001 }

{ "Firstname" : "Mahesh", "salary" : 49500.00000000001 }

{ "Firstname" : "Girish", "salary" : 38500 }

{ "Firstname" : "Abhishek", "salary" : 38500 }

{ "Firstname" : "Sakshi", "salary" : 38500 }

**Delete document of Anay**

> db.Employee.remove({Firstname:"Anay"})

WriteResult({ "nRemoved" : 1 })

> db.Employee.find()

{ "\_id" : ObjectId("638dde0d5b9cae48f3c90016"), "Firstname" : "Ashutosh", "Lastname" :

"Zende", "gender" : "F", "salary" : 33000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638ddef25b9cae48f3c9001a"), "Firstname" : "Smita", "Lastname" : "Patil",

"gender" : "F", "salary" : 27500.000000000004, "deptname" : "Mechanical" }

{ "\_id" : ObjectId("638ddf405b9cae48f3c9001b"), "Firstname" : "Tejas", "Lastname" : "Patil",

"gender" : "M", "salary" : 33000, "deptname" : "Robotics" }

{ "\_id" : ObjectId("638de1655b9cae48f3c9001c"), "Firstname" : "Abhishek", "Lastname" :

"Mane", "gender" : "M", "salary" : 38500, "deptname" : "ENTC" }

{ "\_id" : ObjectId("638de1c05b9cae48f3c9001d"), "Firstname" : "Ashutosh", "Lastname" : "Patil",

"gender" : "M", "salary" : 16500, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de1e75b9cae48f3c9001e"), "Firstname" : "Sakshi", "Lastname" : "Bhosale",

"gender" : "F", "salary" : 38500, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de2245b9cae48f3c9001f"), "Firstname" : "Mahesh", "Lastname" : "Bhosale",

"gender" : "M", "salary" : 49500.00000000001, "deptname" : "Mech" }

{ "\_id" : ObjectId("638de2685b9cae48f3c90021"), "Firstname" : " Prathmesh", "Lastname" : "Shinde",

"gender" : "F", "salary" : 49500.00000000001, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de29e5b9cae48f3c90022"), "Firstname" : "Girish", "Lastname" : "Gaikwad", "gender" : "F", "salary" : 38500, "deptname" : "CSE" } >

**http://beginner-sql-tutorial.com/sql-grant-revoke-privileges-roles.htm**

**SQL GRANT REVOKE Commands**

DCL commands are used to enforce database security in a multiple user database environment. Two types of DCL commands are GRANT and REVOKE. Only Database Administrator's or owner's of the database object can provide/remove privileges on a database object.

**SQL GRANT Command**

SQL GRANT is a command used to provide access or privileges on the database objects to the users.

**The Syntax for the GRANT command is:**

GRANT privilege\_name   
ON object\_name   
TO {user\_name |PUBLIC |role\_name}   
[WITH GRANT OPTION];

* ***privilege\_name*** is the access right or privilege granted to the user. Some of the access rights are ALL, EXECUTE, and SELECT.
* ***object\_name*** is the name of an database object like TABLE, VIEW, STORED PROC and SEQUENCE.
* ***user\_name*** is the name of the user to whom an access right is being granted.
* ***user\_name*** is the name of the user to whom an access right is being granted.
* ***PUBLIC*** is used to grant access rights to all users.
* ***ROLES*** are a set of privileges grouped together.
* ***WITH GRANT OPTION*** - allows a user to grant access rights to other users.

**For Example:** GRANT SELECT ON employee TO user1; This command grants a SELECT permission on employee table to user1.You should use the WITH GRANT option carefully because for example if you GRANT SELECT privilege on employee table to user1 using the WITH GRANT option, then user1 can GRANT SELECT privilege on employee table to another user, such as user2 etc. Later, if you REVOKE the SELECT privilege on employee from user1, still user2 will have SELECT privilege on employee table.

**SQL REVOKE Command:**

The REVOKE command removes user access rights or privileges to the database objects.

The Syntax for the REVOKE command is:

REVOKE privilege\_name   
ON object\_name   
FROM {user\_name |PUBLIC |role\_name}

**For Example:** REVOKE SELECT ON employee FROM user1;This command will REVOKE a SELECT privilege on employee table from user1.When you REVOKE SELECT privilege on a table from a user, the user will not be able to SELECT data from that table anymore. However, if the user has received SELECT privileges on that table from more than one users, he/she can SELECT from that table until everyone who granted the permission revokes it. You cannot REVOKE privileges if they were not initially granted by you.

**Privileges and Roles:**

Privileges: Privileges defines the access rights provided to a user on a database object. There are two types of privileges.

1. **System privileges** - This allows the user to CREATE, ALTER, or DROP database objects.   
   **2) Object privileges** - This allows the user to EXECUTE, SELECT, INSERT, UPDATE, or DELETE data from database objects to which the privileges apply.

Few CREATE system privileges are listed below:

|  |  |
| --- | --- |
| **System Privileges** | **Description** |
| CREATE object | allows users to create the specified object in their own schema. |
| CREATE ANY object | allows users to create the specified object in any schema. |

**The above rules also apply for ALTER and DROP system privileges.**

Few of the object privileges are listed below:

|  |  |
| --- | --- |
| **Object Privileges** | **Description** |
| INSERT | allows users to insert rows into a table. |
| SELECT | allows users to select data from a database object. |
| UPDATE | allows user to update data in a table. |
| EXECUTE | allows user to execute a stored procedure or a function. |

**Roles:**Roles are a collection of privileges or access rights. When there are many users in a database it becomes difficult to grant or revoke privileges to users. Therefore, if you define roles, you can grant or revoke privileges to users, thereby automatically granting or revoking privileges. You can either create Roles or use the system roles pre-defined by oracle.

Some of the privileges granted to the system roles are as given below:

|  |  |
| --- | --- |
| **System Role** | **Privileges Granted to the Role** |
| CONNECT | CREATE TABLE, CREATE VIEW, CREATE SYNONYM, CREATE SEQUENCE, CREATE SESSION etc. |
| RESOURCE | CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER etc. The primary usage of the RESOURCE role is to restrict access to database objects. |
| DBA | ALL SYSTEM PRIVILEGES |

**Creating Roles:**

**The Syntax to create a role is:**

CREATE ROLE role\_name   
[IDENTIFIED BY password];

**For Example:** To create a role called "developer" with password as "pwd", the code will be as follows

CREATE ROLE testing   
[IDENTIFIED BY pwd];

It's easier to GRANT or REVOKE privileges to the users through a role rather than assigning a privilege directly to every user. If a role is identified by a password, then, when you GRANT or REVOKE privileges to the role, you definitely have to identify it with the password.

We can GRANT or REVOKE privilege to a role as below.

**For example:** To grant CREATE TABLE privilege to a user by creating a testing role:

First, create a testing Role

CREATE ROLE testing

Second, grant a CREATE TABLE privilege to the ROLE testing. You can add more privileges to the ROLE.

GRANT CREATE TABLE TO testing;

Third, grant the role to a user.

GRANT testing TO user1;

To revoke a CREATE TABLE privilege from testing ROLE, you can write:

REVOKE CREATE TABLE FROM testing;

**The Syntax to drop a role from the database is as below:**

DROP ROLE role\_name;

**For example:** To drop a role called developer, you can write:

DROP ROLE testing;

***REVOKE Statement Syntax***

REVOKE [ GRANT OPTION FOR ] *privilege\_list*

ON *object*

FROM *user\_name* [ RESTRICT | CASCADE ]

The REVOKE statement takes privileges away from users. The arguments are similar to the GRANT statement. The major difference is the additional RESTRICT or CASCADE keyword and the GRANT OPTION FOR clause. The following describes the optional clauses GRANT OPTION FOR and RESTRICT or CASCADE.

NOTE: If none of the privileges that you are trying to revoke actually exist, an error is raised.

*RESTRICT | CASCADE*

If you use RESTRICT keyword, the privilege will be revoked only from the specified user. If the specified user granted had the WITH GRANT OPTION and granted the same privilege to other users, they will retain the privilege.

If you use CASCADE, it will revoke the privilege and any dependent privileges as a result of your grant. A dependent privilege is one that could exist, if you granted the privilege that you're trying to revoke, which is what you are trying to achieve as a result of your REVOKE statement.

If the optional RESTRICT or CASCADE keywords are not used, PointBase uses RESTRICT by default.

*GRANT OPTION FOR*

If he optional GRANT OPTION FOR clause is used, the WITH GRANT OPTION right is revoked. The actual privilege itself is not revoked. the GRANT OPTION is revoked. CASCADE and RESTRICT may be used in the same way as a normal REVOKE statement.

By the time we finish setting up a Catalog and granting appropriate Privileges to our users, we probably have several thousand Privilege descriptors in INFORMATION\_SCHEMA — more than the count for all other Schema Objects combined. Maintaining them is made easier by the fact that when an Object is dropped, the DBMS will silently destroy all associated Privilege descriptors. That leaves only the problem: how do we adjust for the occasional necessity to remove a Privilege descriptor due to a change in status of a particular User (or Role)? The problem does not occur frequently, but can be mightily cumbersome: the SQL Standard devotes about 40 pages to it. We have managed to simplify the description somewhat, by focussing on the two "essentials" of the process:

1. What we are trying to do is reverse the effects of a GRANT statement, using a REVOKE statement — the clauses of which have almost the same syntax as GRANT's clauses.
2. What we are really doing is deleting Privilege descriptor rows from INFORMATION\_SCHEMA.

The REVOKE statement destroys both Privilege descriptors and Role authorizations and so has two different syntaxes. The first is identified as the <revoke privilege statement> and the second as the <revoke role statement>. The required syntax for the REVOKE statement is:

<revoke privilege statement> ::=REVOKE [ GRANT OPTION FOR ] <privileges> FROM <grantee> [ {,<grantee>}... ]

[ FROM {CURRENT\_USER | CURRENT\_ROLE} ] {RESTRICT | CASCADE}

<revoke role statement> ::=

REVOKE [ ADMIN OPTION FOR ] <Role name> [ {,<Role name>}... ]

FROM <grantee> [ {,<grantee>}... ]

[ FROM {CURRENT\_USER | CURRENT\_ROLE} ]

{RESTRICT | CASCADE}

<grantee> ::= PUBLIC | <AuthorizationID>

The <revoke privilege statement> revokes one or more Privileges on a given Object from one or more grantees, including (possibly) PUBLIC, while the <revoke role statement> revokes the use of one or more Roles from one or more grantees. Only the grantor of the Privileges (or the Roles) may revoke them.

We've already shown you the syntax for the <privileges> Privilege specification; it's used exactly that way in the <revoke privilege statement> form of the REVOKE statement. Here are some examples:

**REVOKE** **SELECT** **ON** **TABLE** Table\_1 **FROM** **PUBLIC** **CASCADE**;

**REVOKE** **INSERT**(column\_1,column\_5) **ON** Table\_1 **FROM** sam **CASCADE**;

**REVOKE** **ALL** **PRIVILEGES** **ON** **TABLE** Table\_1 **FROM** **PUBLIC** **CASCADE**;

**REVOKE** **USAGE** **ON** **DOMAIN** domain\_1 **FROM** bob **CASCADE**;

**REVOKE** **EXECUTE** **ON** **SPECIFIC** **ROUTINE** some\_routine **FROM** sam **CASCADE**;

And here's an example of the <revoke role statement> form of REVOKE:

**REVOKE** assistants\_role **FROM** **PUBLIC** **CASCADE**;

In both cases, if your <grantee> is PUBLIC, you're revoking the Privilege (or the use of the Role) from a list of <grantee>s that contains all of the <AuthorizationID>s in the SQL-environment. If your <grantee> is one or more <AuthorizationID>s, you're revoking the Privilege (or the use of the Role) only from those <AuthorizationID>s. (Remember that an <AuthorizationID> may identify either a User or a Role.)

Remember that, for Tables, GRANT creates Privilege descriptors for both the Table and its Columns. Well, when youREVOKE a Table Privilege, all by-Column Privileges for that Table disappear. The effect is a bit strange; when you revoke a Table Privilege, you lose the Column Privilege (even if it was granted separately) and when you revoke a Column Privilege, you lose that Column Privilege — even if it resulted from a Table Privilege GRANT.

The optional FROM clause names the grantor of the Privileges or the Role you're revoking; CURRENT\_USER is the <AuthorizationID> of the current user and CURRENT\_ROLE is the <AuthorizationID> of the current Role. If you omit the clause, it defaults to FROM CURRENT\_USER — but if CURRENT\_USER is NULL, the clause defaults to FROM CURRENT\_ROLE. If you specify FROM CURRENT\_USER and the current <AuthorizationID> is a <Role name>, or if you specifyFROM CURRENT\_ROLE and the current <Role name> is NULL, the REVOKE statement will fail: your DBMS will return theSQLSTATE error 0L000 "invalid grantor". Here are two examples:

**REVOKE** **UPDATE** **ON** Table\_1 **FROM** sam **FROM** **CURRENT\_USER** **CASCADE**;

-- revokes the UPDATE Privilege on TABLE\_1 from Sam only if the current user

-- granted that Privilege in the first place

**REVOKE** assistants\_role **FROM** **PUBLIC** **FROM** **CURRENT\_ROLE** **CASCADE**;

-- revokes the use of the ASSISTANTS\_ROLE Role from PUBLIC only if the current

-- Role granted the use of that Role in the first place

The optional HIERARCHY OPTION FOR clause (applicable only to the <revoke privilege statement>) allows you to revoke only the WITH HIERARCHY OPTION from the specified SELECT Privilege.

The optional GRANT OPTION FOR (<revoke privilege statement>) and ADMIN OPTION FOR (<revoke role statement>) clauses allow you to revoke only the grantability of a Privilege or a Role. For example, consider these SQL statements:

**GRANT** **UPDATE** **ON** **TABLE** Table\_1 **TO** sam **WITH** **GRANT** **OPTION**;

**REVOKE** **GRANT** **OPTION** **FOR** **UPDATE** **ON** Table\_1 **FROM** sam **CASCADE**;

The first SQL statement allows Sam to update Table\_1, and to pass this Privilege on to others. The second SQL statement revokes the latter ability: Sam can still update Table\_1, but may no longer pass the Privilege on. Here's another example:

**GRANT** assistants\_role **TO** bob **WITH** **ADMIN** **OPTION**;

**REVOKE** **WITH** **ADMIN** **OPTION** **FOR** assistants\_role **FROM** bob **CASCADE**;

The first SQL statement allows Bob to use all of the Privileges belonging to the assistants\_role Role, and to pass the use of this Role on to others. The second SQL statement revokes the latter ability: Bob can still use the Role's Privileges, but may no longer pass that use on.

The GRANT/ADMIN option clauses have another effect. Suppose that a user holds a Privilege on a TableWITH GRANT OPTION, and does so, also with GRANT OPTION. The second user can now do the same for a third user, and so on — for example:

**GRANT** **DELETE** **ON** **TABLE** Sally\_Dates **TO** joe **WITH** **GRANT** **OPTION**;

-- assume Sally does this

**GRANT** **DELETE** **ON** **TABLE** Sally\_Dates **TO** sam **WITH** **GRANT** **OPTION**;

-- assume Joe does this

**GRANT** **DELETE** **ON** **TABLE** Sally\_Dates **TO** bob **WITH** **GRANT** **OPTION**;

-- assume Sam does this

What should happen if Sally now does:

**REVOKE** **DELETE** **ON** **TABLE** Sally\_Dates **FROM** joe;

Here, we've deliberately left off RESTRICT/CASCADE for the sake of the example, so let's assume that the SQL statement works: Joe no longer has the DELETE Privilege on Sally\_Dates. The Privileges handed down from Joe to Sam, and from Sam to Bob, are now called "abandoned Privileges": they are dependent on Joe's DELETE Privilege — and Joe doesn't have it any more. This is where the RESTRICT/CASCADE <keyword>s come in.

* If your REVOKE statement specifies CASCADE, the REVOKE succeeds — and it cascades down to revoke any Privileges that would otherwise be abandoned. In our example, that means both Sam and Bob would no longer have the DELETE Privilege on Sally\_Dates either.
* If your REVOKE statement specifies RESTRICT, the REVOKE succeeds only if the Privilege being revoked has no dependent Privileges. In our example, that means the REVOKE statement would fail.

The same holds true for revoking the use of a Role.

Objects can also become "abandoned" when a Privilege or the use of a Role is revoked. For example, remember that Joe holds the SELECT Privilege on Sally\_Dates and, with this, was able to create his View, JOE\_VIEWS. Now suppose Sally does this:

**REVOKE** **SELECT** **ON** Sally\_Dates **FROM** joe **CASCADE**;

The effect is that, not only does Joe lose his ability to SELECT from Sally\_Dates, but that JOE\_VIEWS is dropped! The reason is that, in effect, JOE\_VIEWS is nothing but a SELECT that Joe does from Sally\_Dates, and since such SELECTs are no longer allowed, the View may no longer exist. If, on the other hand, Sally does the following:

**REVOKE** **SELECT** **ON** Sally\_Dates **FROM** joe **RESTRICT**;

the effect is that the REVOKE statement fails: Sally may not revoke Joe's ability to SELECT from Sally\_Dates because this would mean that JOE\_VIEWS would be abandoned — and this is not allowed. The same holds true for any Object that anyone was able to create only because they held some required Privilege (or were able to use a Role that held that Privilege): if REVOKE ... RESTRICT is used, the statement will fail but if REVOKE ... CASCADE is used, the statement will not only revoke but drop all Objects that would otherwise be abandoned.

If the REVOKE statement isn't able to find a Privilege descriptor for every one of its Privilege specifications, your DBMS will return the SQLSTATE warning 01006 "warning-privilege not revoked".

If you want to restrict your code to Core SQL, don't use the <revoke role statement> form of the REVOKE statement and don't use REVOKE ... CASCADE or the GRANT OPTION FOR or HIERARCHY OPTION FOR clauses. Also, when revoking, make sure that your current <AuthorizationID> is the owner of the Schema that owns the Object you're revoking Privileges for.

**Vertical fragmentation**

* **Create Database :**

mysql> show databases;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| B.Tech\_37 |

| BE\_A\_37 |

| BE\_Harsha |

| BTech\_78 |

| Bank |

| b1\_4 |

| btech39 |

| db\_name |

| horizontal |

| mysql |

| performance\_schema |

| server\_DB |

| sm82 |

| sys |

| vertical |

+--------------------+

16 rows in set (0.12 sec)

mysql> use horizontal;

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Database changed

* **Create Table :**

create table employee(eid int,ename varchar(20),esalary int,ecity varchar(20));

insert into employee values(101,Sakshi,10000,Kolhapur);

insert into employee values(102,Rutika,20000,Solapur);

insert into employee values(103,Pranoti,30000,Pune);

insert into employee values(104,Aarya,40000,Kolhapur);

insert into employee values(105,Teju,50000,Mumbai);

select \* from employee;

mysql> select \* from employee;

+------+---------+---------+----------+

| eid | ename | esalary | ecity |

+------+---------+---------+----------+

| 101 | Sakshi | 10000 | kolhapur |

| 102 | Rutika | 20000 | solapur |

| 103 | Pranoti | 30000 | pune |

| 104 | Aarya | 40000 | kolhapur |

| 105 | Teju | 50000 | mumbai |

+------+---------+---------+----------+

5 rows in set (0.03 sec)

create table empkop(eid int,ename varchar(20),esalary int,ecity varchar(20));

mysql> select \* from empkop;

Empty set (0.00 sec)

**Server.java**

import java.util.\*;

import java.sql.\*;

import java.net.\*;

import java.io.\*;

public class hfs

{

public static void main(String args[])

{

DataInputStream dis;

DataOutputStream dos;

InputStream is;

OutputStream os;

Scanner s1=new Scanner(System.in);

try

{

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/horizontal?useSSL=false","root","root");

Statement st=con.createStatement();

ServerSocket s=new ServerSocket(1520);

Socket s2=s.accept();

is=s2.getInputStream();

os=s2.getOutputStream();

dis=new DataInputStream(is);

dos=new DataOutputStream(os);

int count=0;

String rw=new String(dis.readUTF());

System.out.println(rw);

ResultSet rs1=st.executeQuery(rw);

while(rs1.next())

{

int id=rs1.getInt(1);

String nm=rs1.getString(2);

int sal=rs1.getInt(3);

String ect=rs1.getString(4);

System.out.println(id+"\t"+nm+"\t"+sal+"\t"+ect);

count++;

}

dos.writeInt(count);

String nm;

int id,sal;

String ect;

ResultSet rs2=st.executeQuery(rw);

while(rs2.next())

{

id=rs2.getInt(1); nm=rs2.getString(2);

sal=rs2.getInt(3); ect=rs2.getString(4);

dos.writeInt(id); dos.writeUTF(nm);

dos.writeInt(sal); dos.writeUTF(ect);

}

}

catch(Exception e)

{

System.out.println(e);

}

} }

**Client.java**

import java.util.\*;

import java.sql.\*;

import java.net.\*;

import java.io.\*;

public class hfc

{

public static void main(String args[])

{

DataInputStream dis; DataOutputStream dos;

InputStream is; OutputStream os;

Statement st,st1,st2,st3;

Scanner s1=new Scanner(System.in);

try {

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/horizontal?useSSL=false","root","root");

st=con.createStatement();

Socket s=new Socket("127.0.0.1",1520);

is=s.getInputStream();

os=s.getOutputStream();

dis=new DataInputStream(is);

dos=new DataOutputStream(os);

System.out.println("Enter the query");

String rw=s1.nextLine();

dos.writeUTF(rw);

System.out.println("Horizontal fragmentation");

int id,sal; String nm; String ect;

int count=dis.readInt();

for(int i=0;i<count;i++)

{

id= dis.readInt();

nm=dis.readUTF();

sal= dis.readInt();

ect=dis.readUTF();

st.executeUpdate("insert into empkop values("+id+",'"+nm+"',"+sal+",'"+ect+"')");

}

ResultSet rs1=st.executeQuery("select \* from empkop ");

while(rs1.next())

{ int id1=rs1.getInt(1); nm=rs1.getString(2);

sal=rs1.getInt(3); ect=rs1.getString(4);

System.out.println(id1+"\t"+nm+"\t"+sal+"\t"+ect);

}

}

catch(Exception e)

{ System.out.println(e); }

} }

**output:**

server.java

ubuntu@ubuntu-OptiPlex-3010:~/Desktop/BTech\_78/horizontal$ javac hfs.java

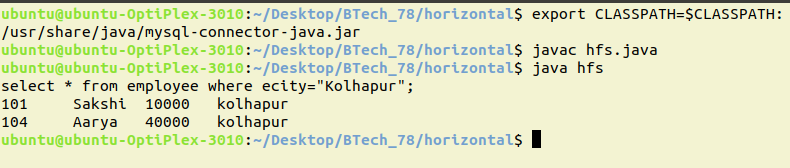
ubuntu@ubuntu-OptiPlex-3010:~/Desktop/BTech\_78/horizontal$ java hfs

select \* from employee where ecity="Kolhapur";

101 Sakshi 10000 kolhapur

104 Aarya 40000 kolhapur

ubuntu@ubuntu-OptiPlex-3010:~/Desktop/BTech\_78/horizontal$



client.java

ubuntu@ubuntu-OptiPlex-3010:~/Desktop/BTech\_78/horizontal$ export CLASSPATH=$CLASSPATH:/usr/share/java/mysql-connector-java.jar

ubuntu@ubuntu-OptiPlex-3010:~/Desktop/BTech\_78/horizontal$ javac hfc.java

ubuntu@ubuntu-OptiPlex-3010:~/Desktop/BTech\_78/horizontal$ java hfc

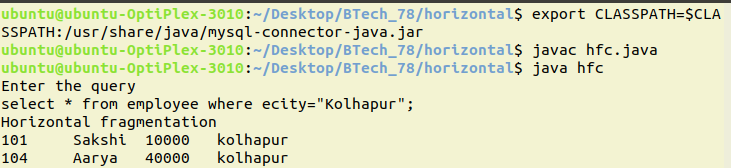
Enter the query

select \* from employee where ecity="Kolhapur";

Horizontal fragmentation

101 Sakshi 10000 kolhapur

104 Aarya 40000 kolhapur



mysql :

mysql> select \* from employee;

+------+---------+---------+----------+

| eid | ename | esalary | ecity |

+------+---------+---------+----------+

| 101 | Sakshi | 10000 | kolhapur |

| 102 | Rutika | 20000 | solapur |

| 103 | Pranoti | 30000 | pune |

| 104 | Aarya | 40000 | kolhapur |

| 105 | Teju | 50000 | mumbai |

+------+---------+---------+----------+

5 rows in set (0.00 sec)

mysql> select \* from empkop;

+------+--------+---------+----------+

| eid | ename | esalary | ecity |

+------+--------+---------+----------+

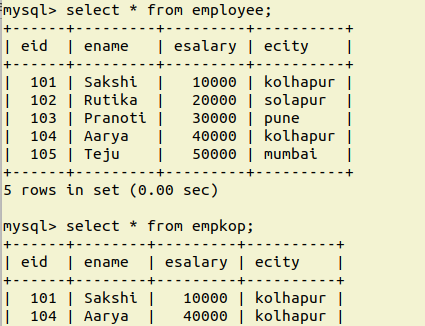
| 101 | Sakshi | 10000 | kolhapur |

| 104 | Aarya | 40000 | kolhapur |

+------+--------+---------+----------+

2 rows in set (0.00 sec)

mysql>



**Assignment 2: Part 1.2 - Horizontal Fragmentation Programs**

**Server-Side Program for Horizontal Fragmentation:**

import java.util.\*;

import java.sql.\*;

import java.net.\*;

import java.io.\*;

public class server

{

public static void main(String args[])

{

DataInputStream dis;

DataOutputStream dos;

InputStream is;

OutputStream os;

Scanner s1=new Scanner(System.in);

try

{

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection("jdbc:mysql://localhost/employee","root","root");

Statement st=con.createStatement();

ServerSocket s=new ServerSocket(1522);

Socket s2=s.accept();

is=s2.getInputStream();

os=s2.getOutputStream();

dis=new DataInputStream(is);

dos=new DataOutputStream(os);

String name;

int id;

int salary;

String city;

int count=0;

String rw=new String(dis.readUTF());

System.out.println(rw);

ResultSet rs1=st.executeQuery(rw);

while(rs1.next())

{

id=rs1.getInt(1);

name=rs1.getString(2);

salary=rs1.getInt(3);

city=rs1.getString(4);

System.out.println(id+"\t"+name+"\t"+salary+"\t"+city);

count++;

}

dos.writeInt(count);

ResultSet rs2=st.executeQuery(rw);

while(rs2.next())

{

id=rs2.getInt(1);

name=rs2.getString(2);

salary=rs2.getInt(3);

city=rs2.getString(4);

dos.writeInt(id);

dos.writeUTF(name);

dos.writeInt(salary);

dos.writeUTF(city);

}

}

catch(Exception e)

{

System.out.println(e);

}

}

}

**Output:**

select \* from emp\_server where emp\_city="Mumbai";

100 Aditi 65000 Mumbai

102 Rahul 35000 Mumbai

104 Anuj 10000 Mumbai

**Client-Side Program for Horizontal Fragmentation:**

import java.util.\*;

import java.sql.\*;

import java.net.\*;

import java.io.\*;

public class client

{

public static void main(String args[])

{

DataInputStream dis;DataOutputStream dos;

InputStream is;OutputStream os;

Statement st,st1,st2,st3;

Scanner s1=new Scanner(System.in);

try

{

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection("jdbc:mysql://localhost/employee","root","root");

st=con.createStatement();

Socket s=new Socket("127.0.0.1",1522);

is=s.getInputStream();

os=s.getOutputStream();

dis=new DataInputStream(is);

dos=new DataOutputStream(os);

System.out.println("Enter the query");

String rw=s1.nextLine();

dos.writeUTF(rw);

System.out.println("Horizontal fragmentation");

int id;

String name;

int salary;

String city;

int count=dis.readInt();

for(int i=0;i<count;i++)

{

id= dis.readInt();

name=dis.readUTF();

salary= dis.readInt();

city=dis.readUTF();

st.executeUpdate("insert into emp\_client values("+id+",'"+name+"',"+salary+",'"+city+"')");

}

ResultSet rs1=st.executeQuery("select \* from emp\_client");

while(rs1.next())

{

int id1=rs1.getInt(1);

name=rs1.getString(2);

salary=rs1.getInt(3);

city=rs1.getString(4);

System.out.println(id1+"\t"+name+"\t"+salary+"\t"+city);

}

}

catch(Exception e)

{

System.out.println(e);

}

}

}

**Output:**

Enter the query

select \* from emp\_server where emp\_city="Mumbai";

Horizontal fragmentation

100 Aditi 65000 Mumbai

102 Rahul 35000 Mumbai

104 Anuj 10000 Mumbai

**Advanced Database Systems**

**Sample Program list**

Q.1 Write Java-MySQL program for horizontal fragmentation (client–server).

Q.2 Write Java-MySQL program for vertical fragmentation (client–server).

Q.3 Write Java-MySQL program for Semijoin.

Q.4 a) Write PLSQL Procedure to find given number is prime or not.

b) Write PLSQL Procedure to find factorial of given number.

Q.5 a) Write PLSQL Procedure to find reverse of given number.

b) Write PLSQL function to find factorial of given number.

Q.6 a) Write PLSQL function to find reverse of given String.

b) Write PLSQL procedure to find average of given numbers.

Q. 7 a) Write PLSQL procedure to find greater from three numbers.

b) Write a cursor that will increase salary by 10% if salary is > 10000 otherwise by 5%.

Q.8 Solve Following Queries on MongoDB

Consider the following

Employee (empid, name, salary, designation)

Create employee collection.

1. Insert at least 5 records with meaningful data.
2. Display all employees.
3. Find the employees whose salary is >10000.
4. Display the employee name whose having salary either 15000 or 25000
5. Update designation of employee 11 to ‘Asst. Prof.’
6. Delete the employees having designation “supervisor”.
7. Display top 2 employees having highest salary.
8. Increment all employee salary by 1000.
9. Increment all employee salary by 6%.

Q.9 Solve Following Queries on MongoDB

Consider the following

Student (rollno, name, marks)

Create student collection.

1. Insert at least 5 records with meaningful data.
2. Display all students.
3. Find the students having marks >70.
4. Display students name having marks between 80 and 90.
5. Update the marks of “Ram” to 90.
6. Delete the students having marks = 75.
7. Display top 2 students having highest marks.
8. Increment all student marks by 5%.
9. Display student name having highest marks.

Q.10 Solve Following Queries on MongoDB

Consider the following

Account (accno, branchname, balance)

Create account collection.

1. Insert at least 6 records with meaningful data.
2. Display all accounts.
3. Find the accounts having balance >25000.
4. Display account numbers having balance between 25000 and 35000.
5. Update the balance of accno. 101 to 35000.
6. Delete the accounts having balance 15000.
7. Display top 2 accounts having highest balance.
8. Display total balance for each branch.
9. Display branch name having balance 20000.

Q.11 Consider the following collection

Student (rollno, name, marks, deptname, city)

Solve following Queries using MongoDB

* + 1. Display document of roll no. 1
    2. Display marks of “Ram”.
    3. Display student having highest marks.
    4. Display students having marks greater than 70.
    5. Modify city of “Krishna” to “Pune”.
    6. Display maximum marks of each department.
    7. Display city of students having roll no. 10 to 20.
    8. Display student collection in descending order of marks.
    9. Delete document of roll no. 5.

Q.12 Consider the following collection

Employee (empid, name, salary, deptname)

Solve following Queries using MongoDB

1. Find employees of CSE departname.
2. Find the name of employees having salary 20000.
3. Find salary of ‘Ram”
4. Find department name of ‘Krishna”
5. Update department of Krishna to ETC.
6. Increment salary by 1000
7. Delete document of “Ram”
8. Find total employees of each department.

Q.13 Consider the following collection

Employee (empid, name, salary, deptname)

Solve following Queries using MongoDB

1. Find the name of employees having salary 25000.
2. Display employees in descending order of salary.
3. Display top 2 employees having highest salary.
4. Find maximum salary.
5. Increment all employee salary by 6%.
6. Display employee name and salary
7. Delete documents of Chemical department.
8. Display salary and department name of “Ram”.

Q.14

a)

1. create table locations(locid varchar(5),city varchar(10),state varchar(5),country varchar(10));
2. create table sales(pid varchar(5),timeid varchar(5),locid varchar(5),sales varchar(5));
3. create table products(pid varchar(5), pname varchar(10), category varchar(15),price varchar(5));
4. create table time(timeid varchar(5),year varchar(5));

Insert appropriate data and perform OLAP Rollup and CUBE operators.

b) Demonstrate Grant and Revoke commands

Q.15 Solve Following Queries on MongoDB

Consider the following

Student (rollno, name, marks)

1. Display document of "Ram"
2. Marks greater than 80 and less than 90
3. Display name and marks of students having marks equal or greater than 80 and < 90
4. Display name and marks of student having rollno. 1
5. Display documents in ascending order of name
6. Display first 2 documents
7. Display document of Ram or rollno 3
8. Display document of Ram and rollno 1
9. Display all in documents in descending order of Marks
10. Display name and marks of student having highest marks

Q.16 Solve Following Queries on CouchDB

Consider the following

Employee (empid, name, salary, deptname)

1. Display document of "Ram"
2. Display salary greater than 20000 and less than 30000
3. Display name and department name of employees having salary equal or greater than 10000 and < 15000
4. Display name and salary of employee having empid 1
5. Display documents in ascending order of name
6. Display first 2 documents
7. Display document of Ram or empid. 3
8. Display all in documents in descending order of salary
9. Display name and salary of employee having highest salary.
10. Display department having highest salary.

**Question Bank for Oral exam**

**Unit 1: Parallel and Distributed Databases**

1. What is a parallel system?

2. What is throughput and response time? Or how to measure performance of database systems?

3. What is Speedup and Scaleup? Explain linear and sublinear speedup and scaleup.

4. Explain parallel database architectures. Give their advantages and disadvantages.

5. What is distributed database system? Explain its advantages and disadvantages.

6. Explain different partitioning techniques used in parallel database systems.

7. Compare different partitioning techniques used in parallel database systems.

8. What is skew? How to handle it?

9. What is interquery and intraquery parallelism?

10. What is interoperation and intraoperation parallelism?

11. What is pipelined parallelism and independent parallelism?

12. Explain how to store data in distributed database system.

13. Explain replication, its types, and advantages, disadvantages.

14. Explain horizontal and vertical fragmentation.

15. Explain 2 phase commit (2PC) protocol.

16. How 2PC protocol handles failures?

17. Explain 3 phase commit (3PC) protocol.

18. Explain Semijoin in detail.

19. What is local transaction and global transaction?

20. What is transaction manager and transaction coordinator?

21. What are ACID properties?

**Unit 2: Advanced SQL**

1. What is different SQL character, number functions?

2. What is view, relation, materialized views, primary key, foreign key, unique, check and not null constraint?

3. What are different DDL and DML commands? Explain it with syntax and example.

4. What is PLSQL? Why it is used? Explain PLSQL block.

5. What is stored procedure? What are its advantages?

6. What is PLSQL function?

7. Give difference between Procedure and function.

8. What are in, out and inout parameters?

9. Write PLSQL procedure and function for factorial, prime number, reverse number, greater from 3 numbers.

10. What is trigger? Explain its types and applications?

11. What is cursor? Explain its types and attributes?

12. Define embedded and dynamic SQL?

**Unit 3: NoSQL Database Management**

1. What is NoSQL? Explain its features, advantages and disadvantages.

2. Explain types of NoSQL databases with example.

3. What is MongoDB? Explain its features, advantages and disadvantages.

4. Explain all MongoDB commands/functions used for creating collections, retrieving, updating,

deleting, sorting, limit, inserting, aggregating documents with examples.

5. What is CouchDB? Explain its features, advantages and disadvantages.

6. Give difference between MongoDB and CouchDB.

**Unit 4: Database Administration and Security**

1. What is need for and role of databases in an organization?

2. What is DBA? What are its responsibilities?

3. What is DA? What are its responsibilities?

4. Explain DBA’s managerial role.

5. Explain DBA’s technical role.

6. Explain database administration tools- Data dictionary and CASE tools.

7. Describe data dictionary.

8. Describe CASE tools.

9. What are security vulnerabilities and its measures?

10. Describe grant and revoke commands.

**Unit 5: Business Intelligence and Data Warehouse**

1. What is data, information, knowledge?

2. What is decision support system?

3. What are OLTP and OLAP?

4. What is data warehouse? Give its characteristics.

5. What is Data Mart? Give example.

6. Explain types of OLAP.

7. Explain Star Schema, Snowflake and fact constellation schema.

8. What is Fact table and dimension table?

9. Explain OLAP operations.

10. Explain materialized views.

11. Explain Rollup and Cube operator.

12. What is Business Intelligence? Draw Business Intelligence architecture.

**Unit 6: Data Mining**

1. What is Data Mining?

2. What are the different data mining tasks? Explain each of them.

3. What is KDD? Explain steps performed in KDD.

4. What are the different data mining issues?

5. What are the different data mining algorithms?

**Assignment No. 8**

Java-MongoDB Connectivity and CRUD Operations in MongoDB using Java

Just click on the following link to download the required driver:

<https://search.maven.org/remotecontent?filepath=org/mongodb/mongo-java-driver/2.12.3/mongo-java-driver-2.12.3.jar>

To export the required driver’s packages and classes in program:

**export CLASSPATH=$CLASSPATH:/home/ubuntu/Downloads/mongo-java-driver-2.12.3.jar**

**Program:**

import com.mongodb.BasicDBObject;

import com.mongodb.DB;

import com.mongodb.DBCollection;

import com.mongodb.DBCursor;

import com.mongodb.DBObject;

import com.mongodb.MongoClient;

import com.mongodb.WriteResult;

public class db1 {

public static void main(String[] args) throws Exception {

try{

MongoClient mongo = new MongoClient("localhost" , 27017);

DB db = mongo.getDB("dypcet");

//Create and Insert Operations

System.out.println("---Create and Insert Operations---\n");

System.out.println("Collection created successfully.");

DBCollection col=db.getCollection("student");

BasicDBObject b1 = new BasicDBObject();

b1.put("first\_name", "Sohail");

b1.put("last\_name","Shaikh");

b1.put("gender","Male");

b1.put("department","CSE");

col.insert(b1);

BasicDBObject b2 = new BasicDBObject();

b2.put("first\_name", "Vicky");

b2.put("last\_name","Singh");

b2.put("gender","Male");

b2.put("department","ENTC");

col.insert(b2);

BasicDBObject b3 = new BasicDBObject();

b3.put("first\_name", "Anuj");

b3.put("last\_name","Patil");

b3.put("gender","Male");

b3.put("department","Civil");

col.insert(b3);

System.out.println("Document inserted Successfully.\n");

//Retrieve Operation

System.out.println("---Retrieve Operation---\n");

DBCursor cursor = col.find();

while (cursor.hasNext())

{

System.out.println(cursor.next());

}

//Update Operation

System.out.println("\n---Update Operation---\n");

BasicDBObject query = new BasicDBObject();

query.put("department", "Civil");

BasicDBObject newDocument = new BasicDBObject();

newDocument.put("department", "Chemical");

BasicDBObject updateObject = new BasicDBObject();

updateObject.put("$set", newDocument);

db.getCollection("student").update(query, updateObject);

System.out.println("Document Updated Successfully.\n");

DBCursor cursor1 = col.find();

while (cursor1.hasNext())

{

System.out.println(cursor1.next());

}

//Delete Operation

System.out.println("\n---Delete Operation---\n");

BasicDBObject query1 = new BasicDBObject();

query1.put("first\_name", "Anuj");

db.getCollection("student").remove(query1);

System.out.println("Document Deleted Successfully.\n");

DBCursor cursor2 = col.find();

while (cursor2.hasNext())

{

System.out.println(cursor2.next());

}

}

catch(Exception e)

{

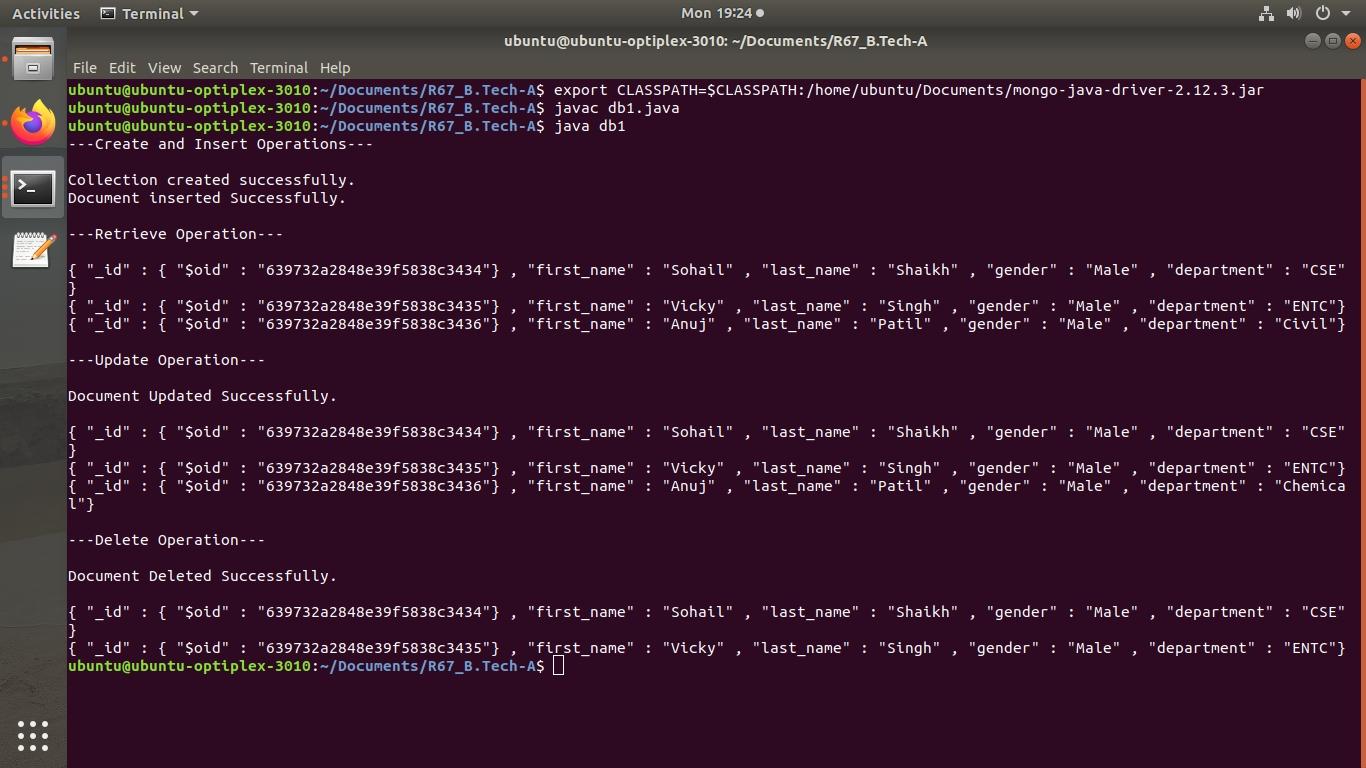
System.out.println(e);

}

}

}

**Output:**

****

**Assignment 4: Oracle Sequence and Synonym**

**-- Sequence Example**

SQL> create sequence productid start with 1 nocache;

Sequence created.

SQL> create sequence billno start with 1 nocache;

Sequence created.

SQL> create table product (productid number(5), Prodname varchar2(10), Prize number(5));

Table created.

SQL> insert into product values( productid.nextval, 'Pen', 10);

1 row created.

SQL> insert into product values( productid.nextval, 'Pencil',5);

1 row created.

SQL> select \* from product;

PRODUCTID PRODNAME PRIZE

---------- ---------- ----------

1 Pen 10

2 Pencil 5

SQL> insert into product values( productid.currval, 'notebook',50);

1 row created.

SQL> select \* from product;

PRODUCTID PRODNAME PRIZE

---------- ---------- ----------

1 Pen 10

2 Pencil 5

2 notebook 50

**--Synonym Example**

SQL> create table student(id number, roll\_no number, name varchar(10), dept varchar(10));

Table created.

SQL> insert into student values(1,82,'Shreya','CSE');

1 row created.

SQL> insert into student values(2,83,'Mayur','Civil');

1 row created.

SQL> insert into student values(3,84,'Atharva','ENTC');

1 row created.

SQL> create public synonym stud for student;

Synonym created.

SQL> select \* from stud;

ID ROLL\_NO NAME DEPT

---------- ---------- ---------- ----------

1 82 Shreya CSE

2 83 Mayur Civil

3 84 Atharva ENTC

SQL> select \* from stud where dept='CSE';

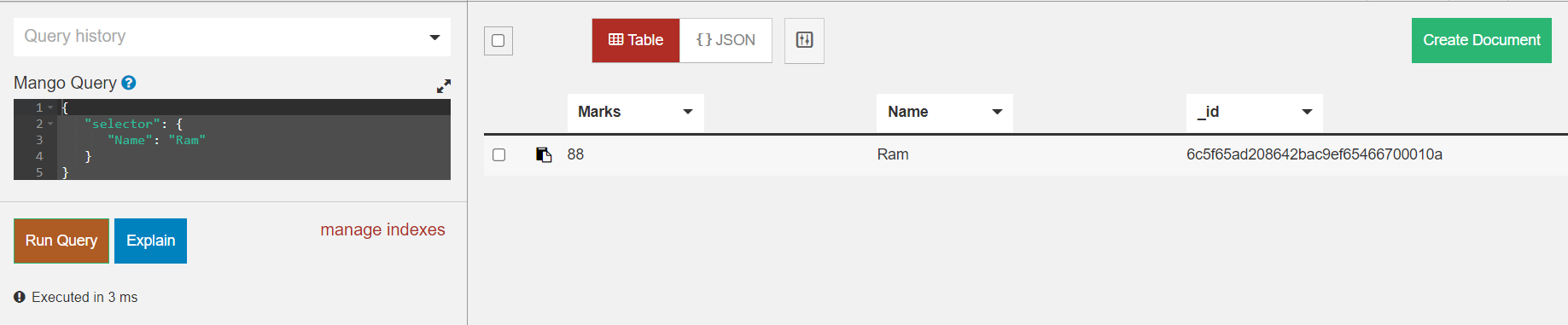
ID ROLL\_NO NAME DEPT

---------- ---------- ---------- ----------

1 82 Shreya CSE

Assignment 9: CouchDB Queries:

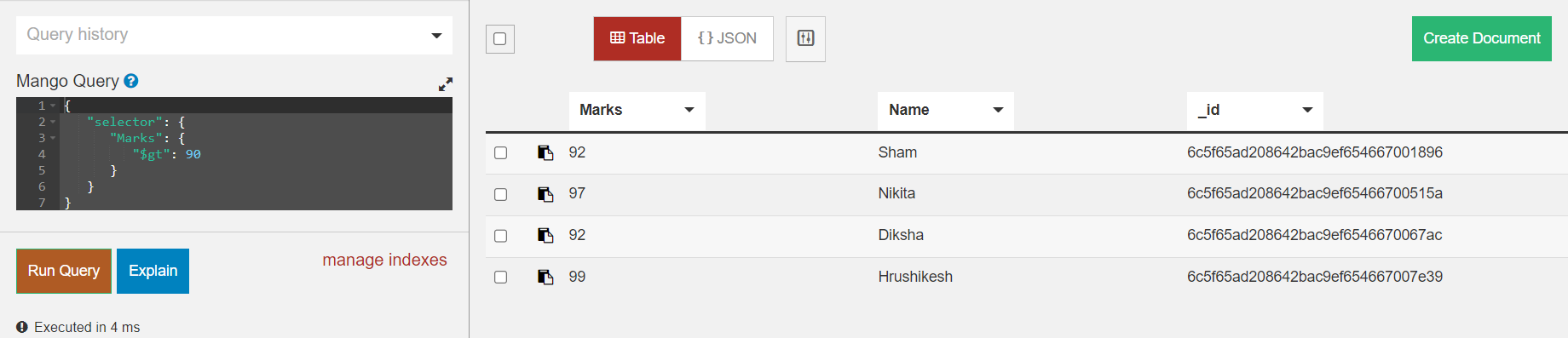
1. Display Document of Ram:



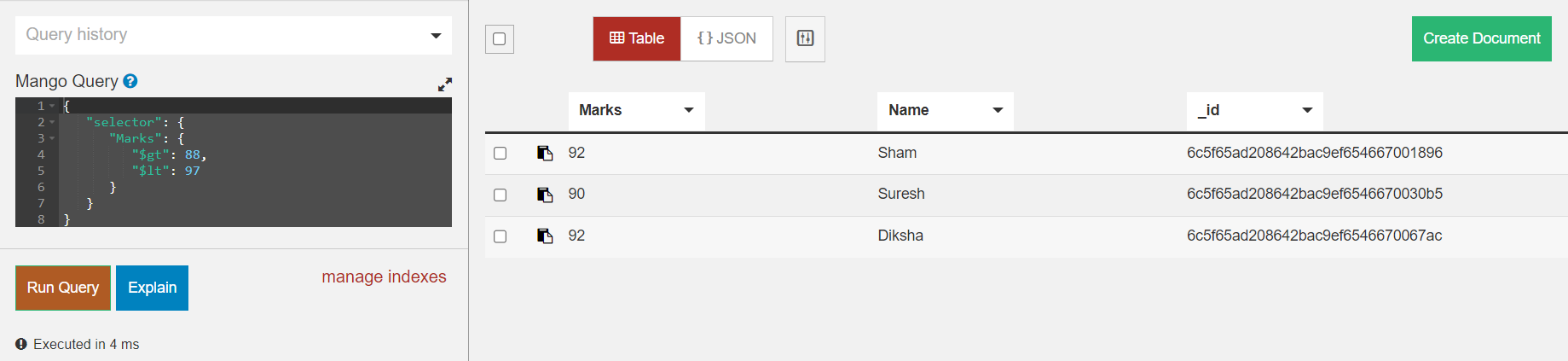
1. Marks Equal to 90:



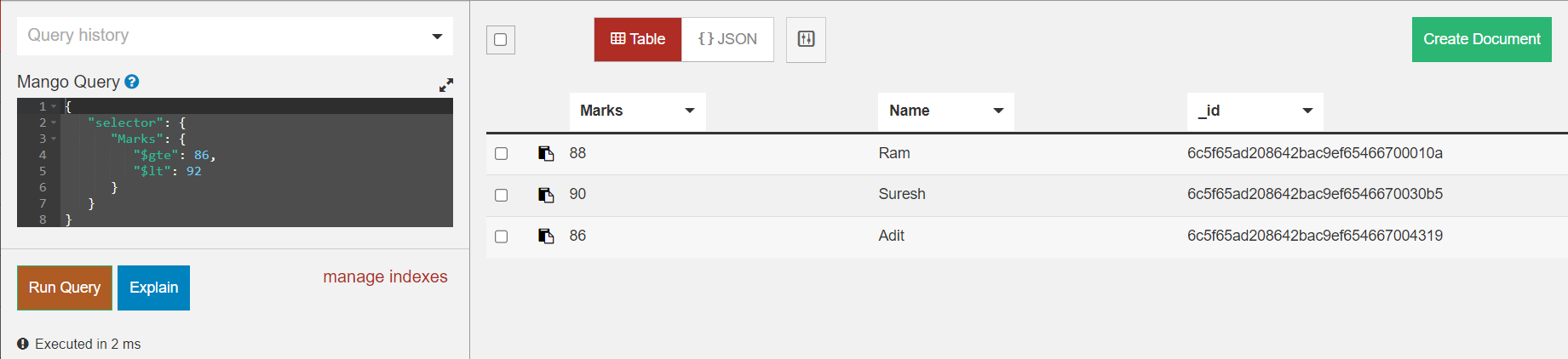
1. Marks greater than 90:



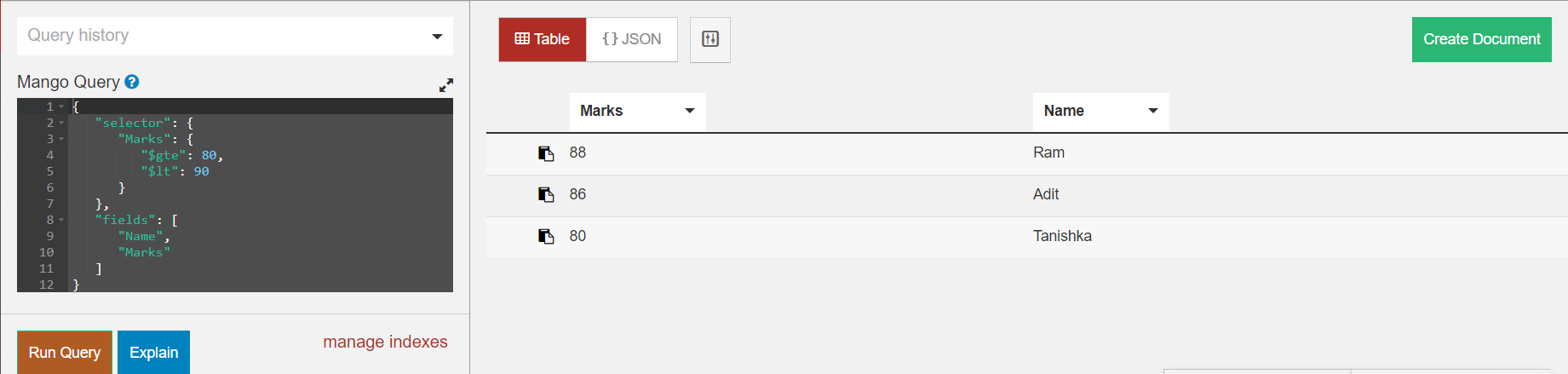
1. Marks greater than 88 and less than 97:



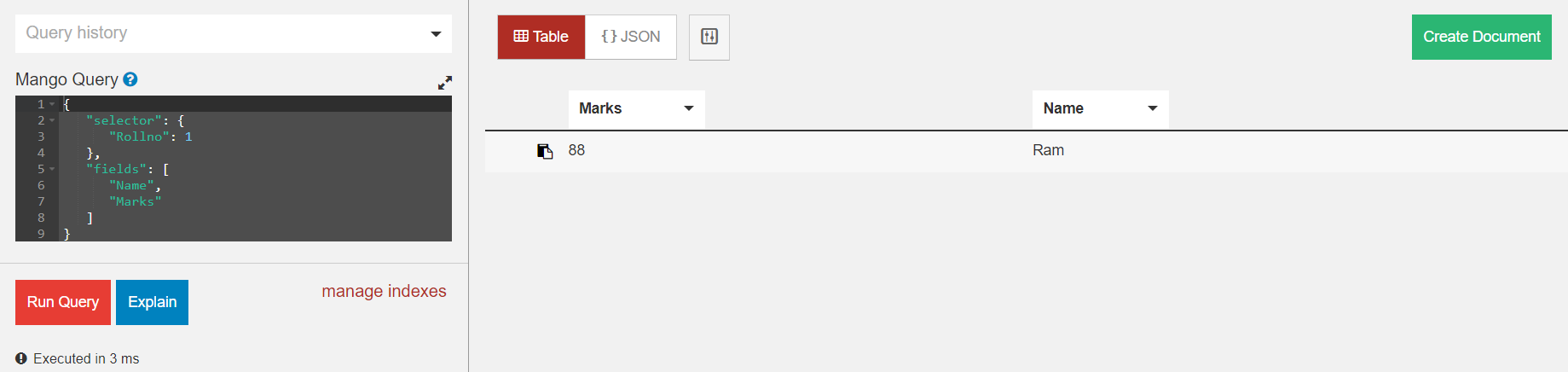
1. Marks equal to or greater than 86 and less than 92:



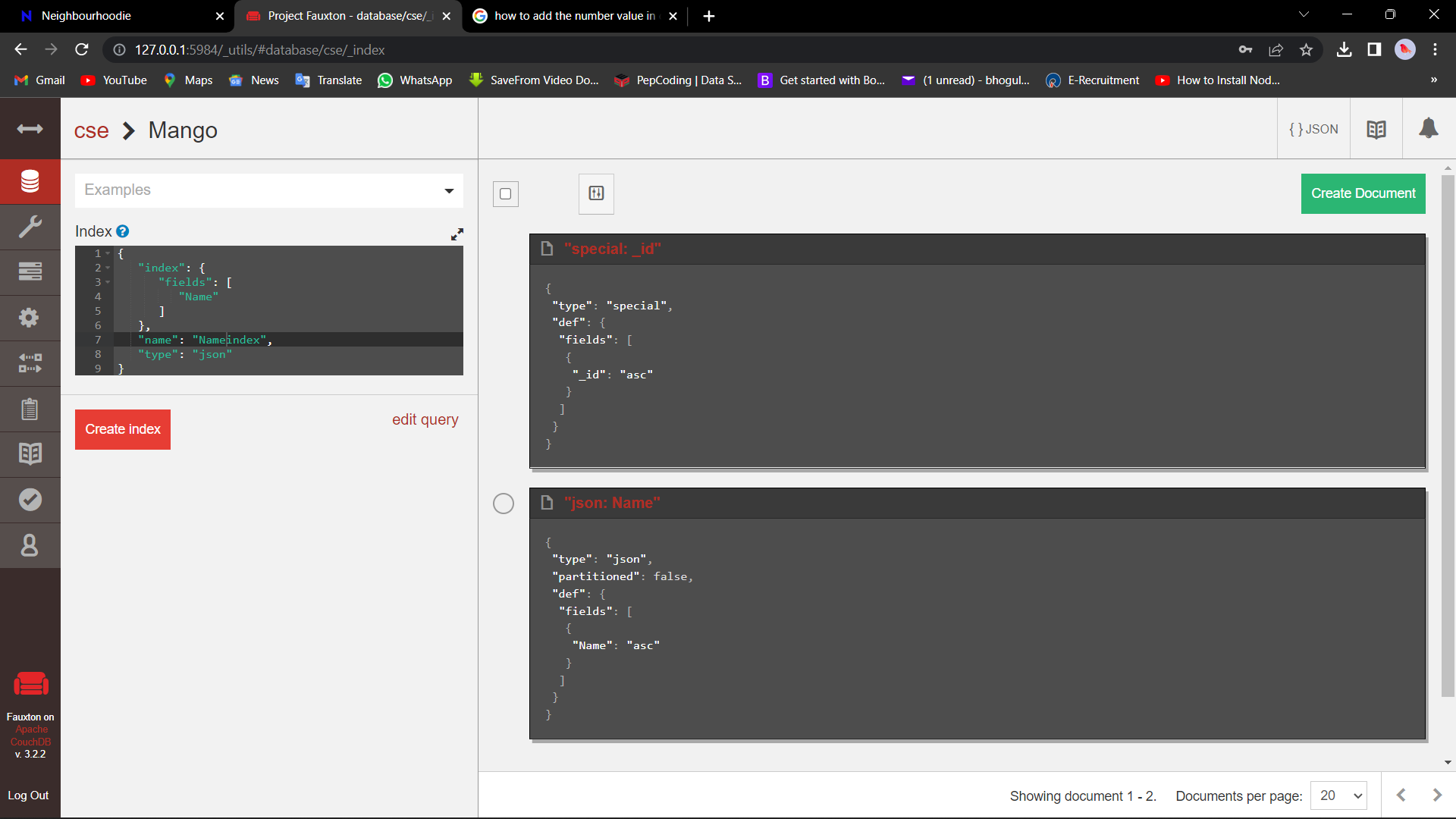
1. Display names and marks of students having marks equal to or greater than 80 and less than 90:



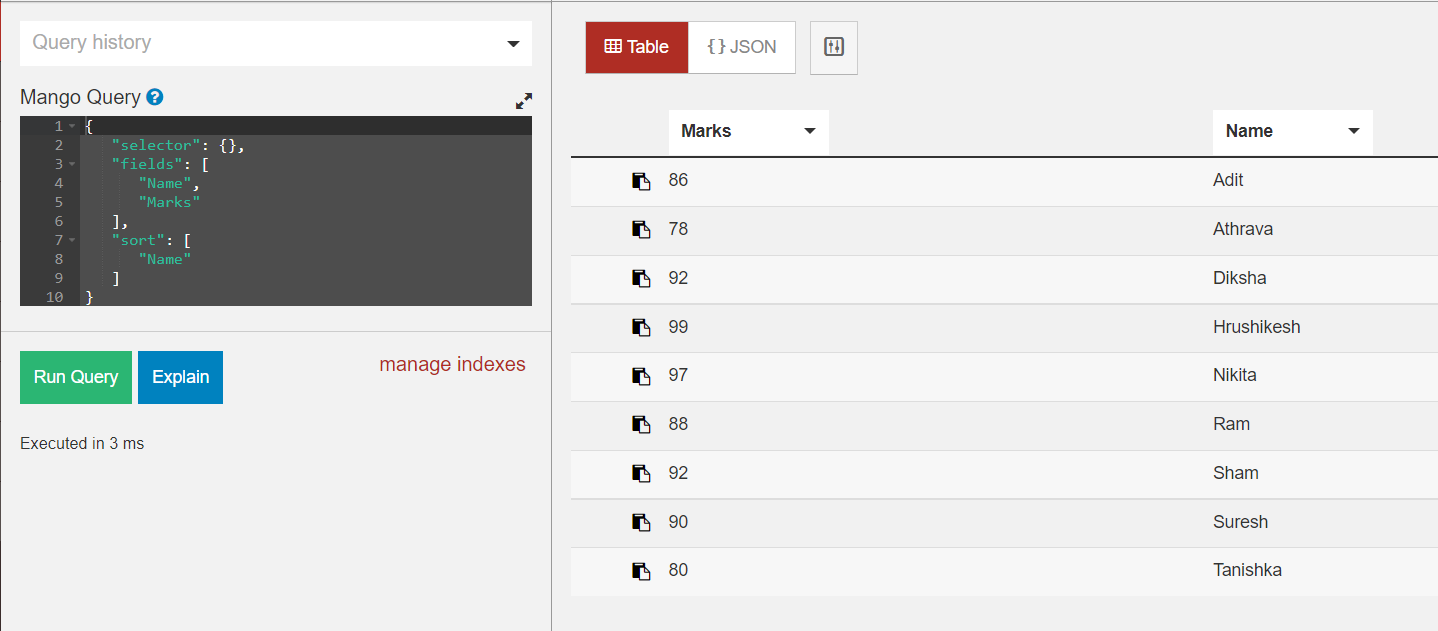
1. Display the name and mark of the student having Roll no 1:

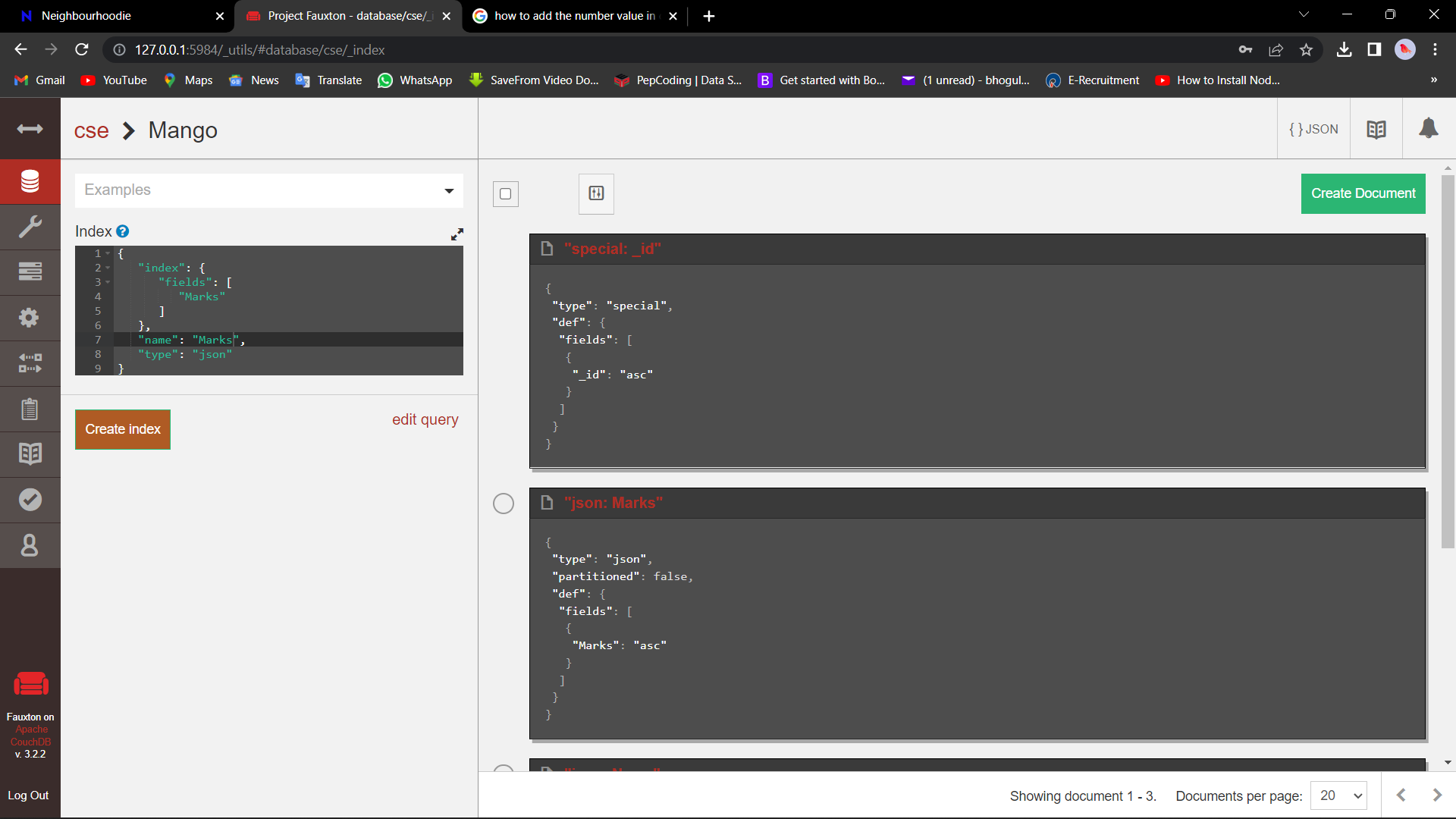


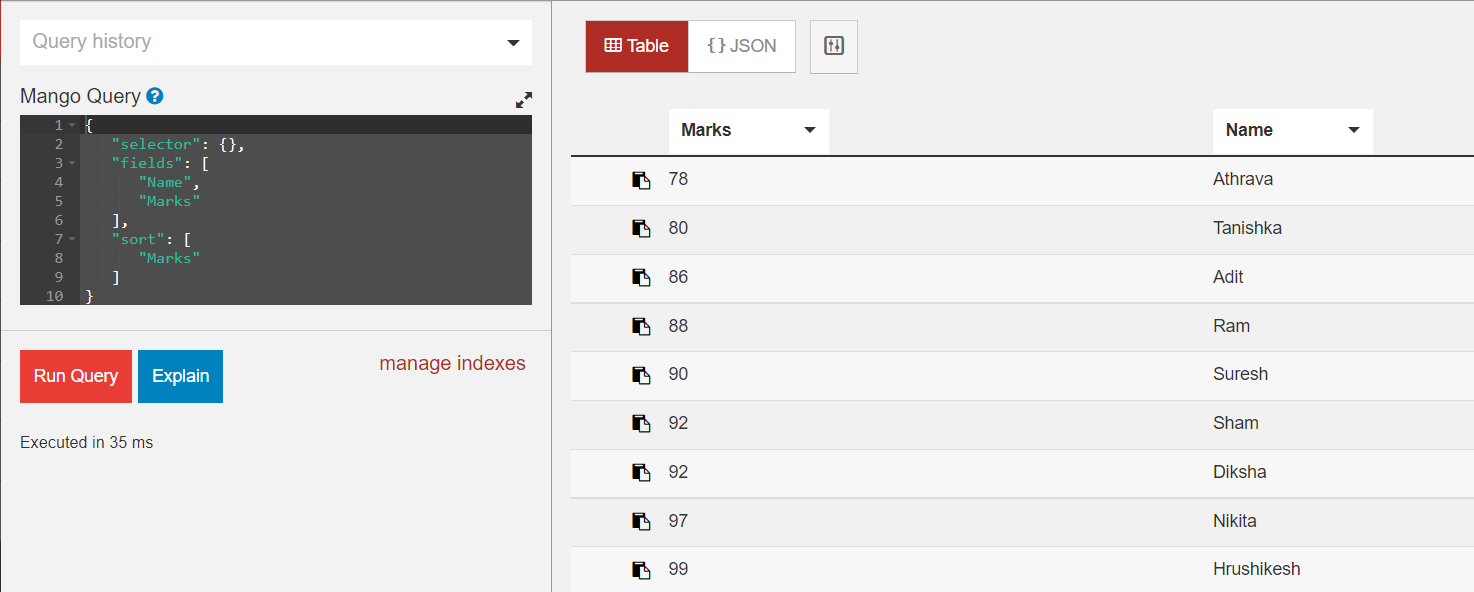
1. Create an index on the name:



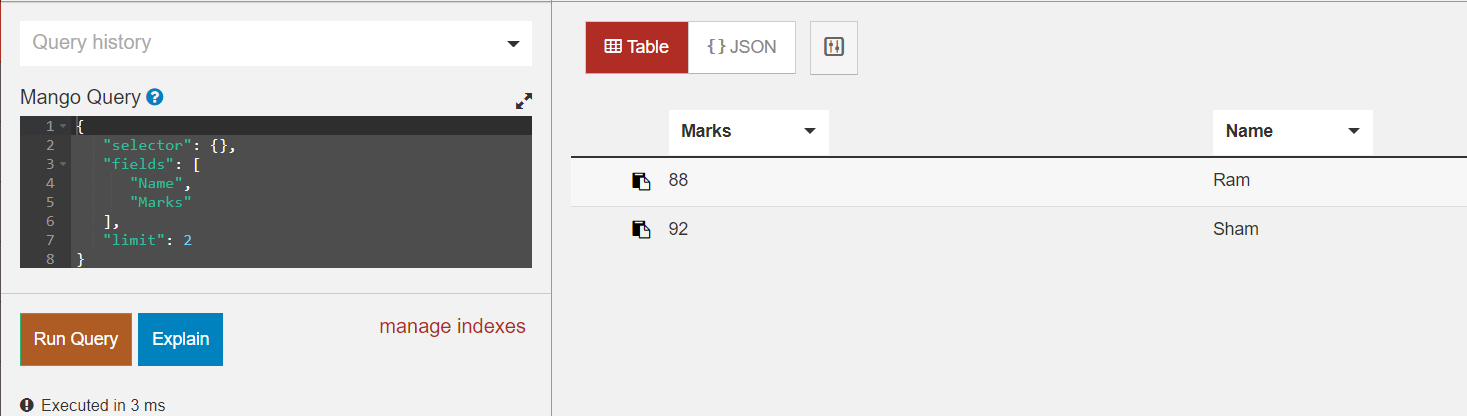
1. Display documents in ascending order of name:



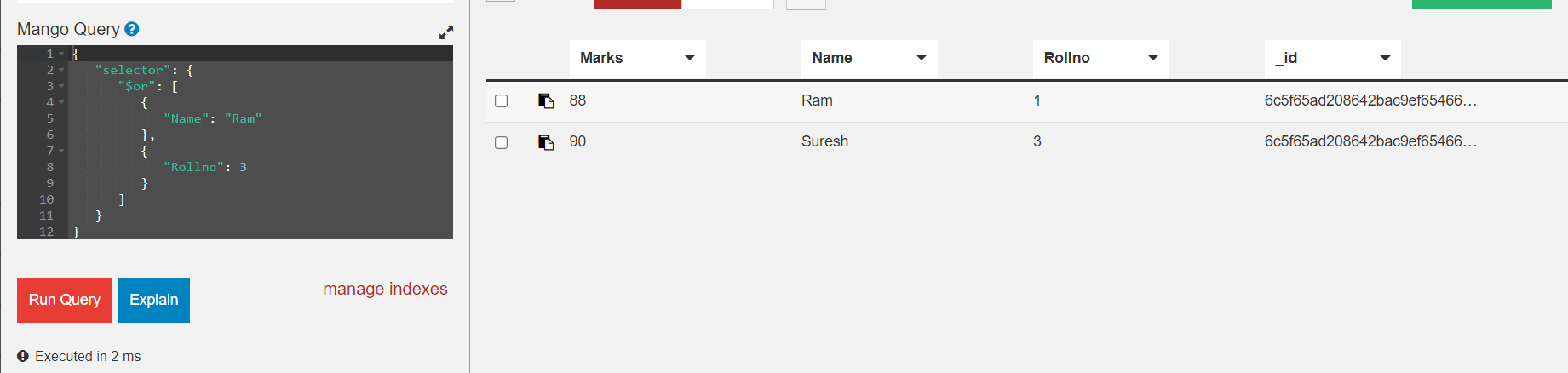
1. Create an index on marks: 
2. Display documents in ascending order of marks:



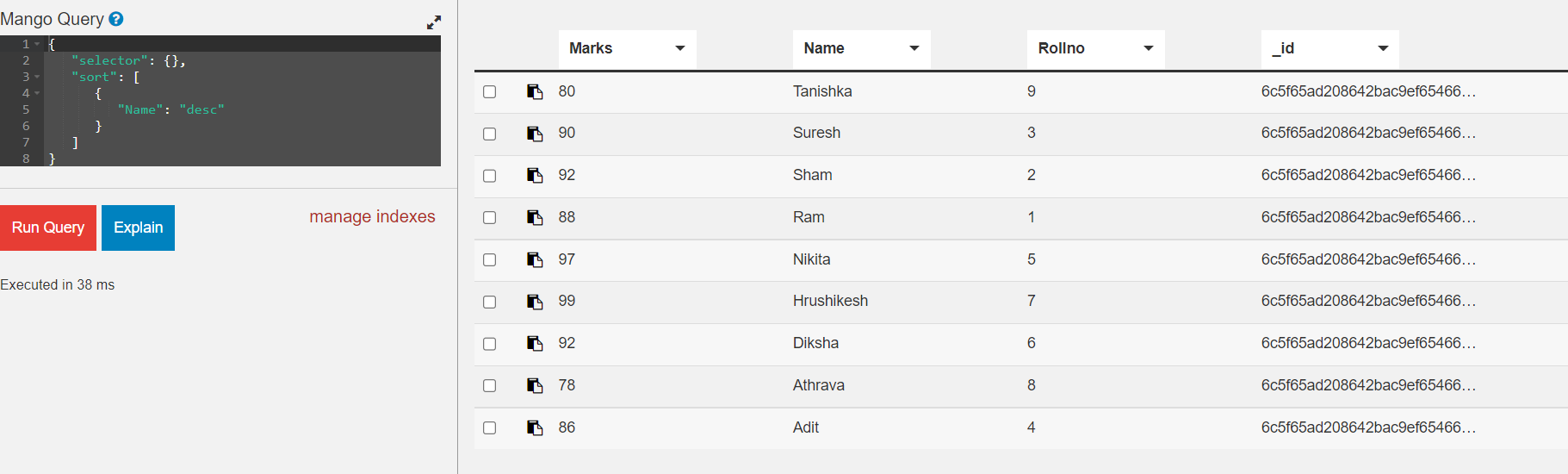
1. Display the first 2 documents:



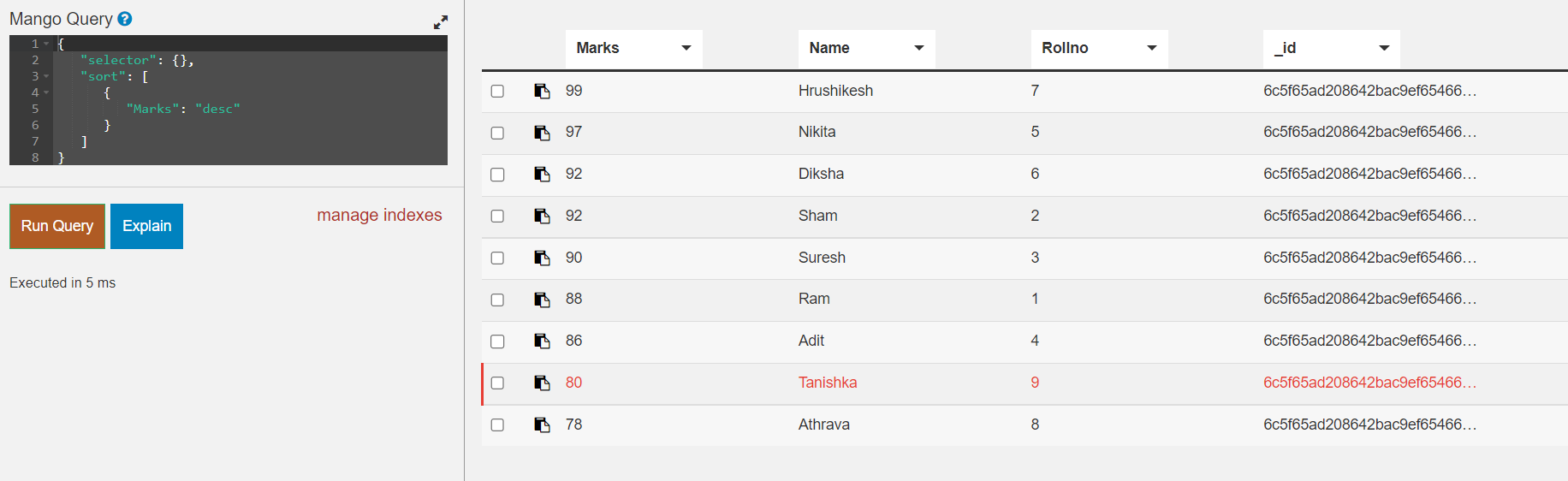
1. Display the document of Ram or Roll no 3:



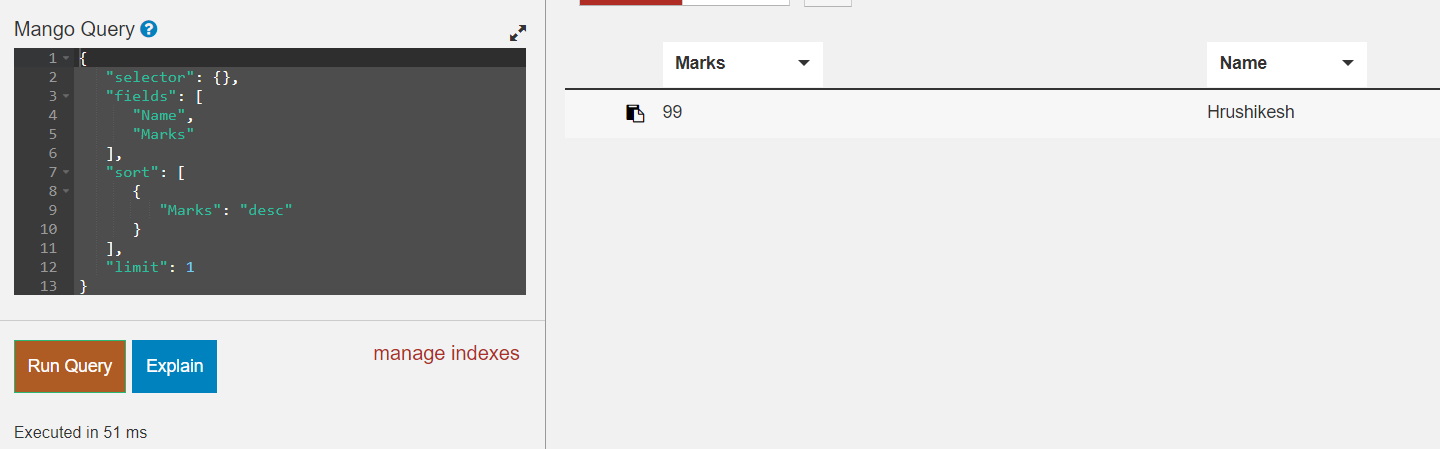
1. Display all the documents in descending order of name:



1. Display all the documents in descending order of the marks:



1. Display names and marks of students having the highest marks:



**Assignment 3: Part 1.2 - Semijoin operation program**

import java.sql.\*;

class bank

{

public static void main(String args[])

{

int acc\_no=0;

String cust\_name;

String branch;

float balance;

try

{

Class.forName("com.mysql.jdbc.Driver");

Connection con;

con=DriverManager.getConnection("jdbc:mysql://localhost:3306/DYP\_Bank?useSSL=false","root","root");

Statement stmt1=con.createStatement();

Statement stmt2=con.createStatement();

Statement stmt3=con.createStatement();

Statement stmt4=con.createStatement();

ResultSet rs=stmt1.executeQuery("select acc\_no from depositor");

while(rs.next())

{

acc\_no= rs.getInt(1);

stmt2.executeUpdate("insert into temp1 values("+acc\_no+");");

}

ResultSet rs1=stmt1.executeQuery("select \* from account natural join temp1");

while(rs1.next())

{

acc\_no= rs1.getInt(1);

branch=rs1.getString(2);

balance=rs1.getFloat(3);

stmt3.executeUpdate("insert into temp2 values("+acc\_no+",'"+branch+"',"+balance+");");

}

ResultSet rs2=stmt1.executeQuery("select \* from depositor natural join temp2");

while(rs2.next())

{

acc\_no= rs2.getInt(1);

cust\_name=rs2.getString(2);

branch=rs2.getString(3);

balance=rs2.getFloat(4);

stmt4.executeUpdate("insert into resultset values("+acc\_no+",'"+cust\_name+"','"+branch+"',"+balance+");");

}

System.out.println("Semijoin Result:");

ResultSet rs3 = stmt1.executeQuery("select \* from resultset");

while(rs3.next()){

acc\_no=rs3.getInt(1);

cust\_name=rs3.getString(2);

branch=rs3.getString(3);

balance=rs3.getFloat(4);

System.out.println(acc\_no+"\t"+cust\_name+"\t"+branch+"\t"+balance);

}

con.close();

}

catch(Exception e){ System.out.println(e);

}

}

}

**Output:**

Semijoin Result:

120 Aditi Rajarampuri 10000.0

122 Vicky Shahupuri 20000.2

123 Ram Tarabai Park 25000.3

**Experiment No : 01**

**MYSQL Installation on Ubuntu**

sudo apt update

sudo apt upgrade

sudo apt install mysql-server

mysql --version

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Set Password**

sudo mysql\_secure\_installation

password- root

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Enter in Mysql**

sudo mysql -u root

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Solve following Queries:**

**Consider the following schema :**

employee(empid, empname, salary, designation, company-name)

**Create database Company :**

mysql> create database company;

Query OK, 1 row affected (0.12 sec)

mysql> show databases;

+--------------------------+

| Database |

+--------------------------+

| Bank |

| Customers |

| Horizontal\_Fragmentation |

| company |

| employee |

| exp9 |

| information\_schema |

| mysql |

| performance\_schema |

| sys |

+--------------------------+

10 rows in set (0.06 sec)

**Execute the following queries:**

**1) Create employee table :**

mysql> use company;

Database changed

create table employee(empid int,empname varchar(20),salary float,designation varchar(20),company\_name varchar(20));

Query OK, 0 rows affected (0.93 sec)

**2) Add city column into employee :**

mysql> alter table employee add(city varchar(25));

Query OK, 0 rows affected (0.39 sec)

Records: 0 Duplicates: 0 Warnings: 0

**3) Insert at least 10 records with meaningful data.**

mysql> insert into employee values(101,"Sakshi",50000,"CEO","TCS","Pune");

Query OK, 1 row affected (0.22 sec)

mysql> insert into employee values(102,"Chirag",50000,"CEO","Tech Mahindra","Pune");

Query OK, 1 row affected (0.12 sec)

mysql> insert into employee values(103,"Shivani",40000,"Manager","KPIT","Mumbai");

Query OK, 1 row affected (0.12 sec)

mysql> insert into employee values(104,"Ashu",40000,"Manager","DXC","Kolhapur");

Query OK, 1 row affected (0.12 sec)

mysql> insert into employee values(105,"Swara",20000,"HR","DXC","Solapur");

Query OK, 1 row affected (0.13 sec)

mysql> insert into employee values(106,"Swayam",21000,"HR","VOIS","Kolhapur");

Query OK, 1 row affected (0.15 sec)

mysql> insert into employee values(107,"Sara",9000,"Supervisor","Tech Mahindra","Pune");

Query OK, 1 row affected (0.10 sec)

mysql> insert into employee values(108,"Swati",30000,"employee","Cognizant","Mumbai");

Query OK, 1 row affected (0.15 sec)

mysql> insert into employee values(109,"Ram",30000,"employee","DXC","Mumbai");

Query OK, 1 row affected (0.10 sec)

mysql> insert into employee values(110,"Piyush",14000,"employee11","TCS","Pune");

Query OK, 1 row affected (0.14 sec)

mysql> select \* from employee;

+-------+----------+--------+-------------+---------------+----------+

| empid | empname | salary | designation | company\_name | city |

+-------+----------+--------+-------------+---------------+----------+

| 101 | Sakshi | 50000 | CEO | TCS | Pune |

| 102 | Chirag | 50000 | CEO | Tech Mahindra | Pune |

| 103 | Shivani | 40000 | Manager | KPIT | Mumbai |

| 104 | Ashu | 40000 | Manager | DXC | Kolhapur |

| 105 | Swara | 20000 | HR | DXC | Solapur |

| 106 | Swayam | 21000 | HR | VOIS | Kolhapur |

| 107 | Sara | 9000 | Supervisor | Tech Mahindra | Pune |

| 108 | Swati | 30000 | employee | Cognizant | Mumbai |

| 109 | Ram | 30000 | employee | DXC | Mumbai |

| 110 | Piyush | 14000 | employee11 | TCS | Pune |

+-------+----------+--------+-------------+---------------+----------+

10 rows in set (0.00 sec)

**4) Display all employees with their salary :**

mysql> select empname,salary from employee;

+----------+--------+

| empname | salary |

+----------+--------+

| Sakshi | 50000 |

| Chirag | 50000 |

| Shivani | 40000 |

| Ashu | 40000 |

| Swara | 20000 |

| Swayam | 21000 |

| Sara | 9000 |

| Swati | 30000 |

| Ram | 30000 |

| Piyush | 14000 |

+----------+--------+

10 rows in set (0.00 sec)

**5) Find the name of the employee along with their id :**

mysql> select empid,empname from employee;

+-------+----------+

| empid | empname |

+-------+----------+

| 101 | Sakshi |

| 102 | Chirag |

| 103 | Shivani |

| 104 | Ashu |

| 105 | Swara |

| 106 | Swayam |

| 107 | Sara |

| 108 | Swati |

| 109 | Ram |

| 110 | Piyush |

+-------+----------+

10 rows in set (0.00 sec)

**6) Find employees who are living in “Kolhapur” :**

mysql> select \* from employee where city="Kolhapur";

+-------+----------+--------+-------------+--------------+----------+

| empid | empname | salary | designation | company\_name | city |

+-------+----------+--------+-------------+--------------+----------+

| 104 | Ashu | 40000 | Manager | DXC | Kolhapur |

| 106 | Swayam | 21000 | HR | VOIS | Kolhapur |

+-------+----------+--------+-------------+--------------+----------+

2 rows in set (0.00 sec)

**7) Find name of the employees whose salary is >10000 :**

mysql> select \* from employee where salary>10000;

+-------+----------+--------+-------------+---------------+----------+

| empid | empname | salary | designation | company\_name | city |

+-------+----------+--------+-------------+---------------+----------+

| 101 | Sakshi | 50000 | CEO | TCS | Pune |

| 102 | Chirag | 50000 | CEO | Tech Mahindra | Pune |

| 103 | Shivani | 40000 | Manager | KPIT | Mumbai |

| 104 | Ashu | 40000 | Manager | DXC | Kolhapur |

| 105 | Swara | 20000 | HR | DXC | Solapur |

| 106 | Swayam | 21000 | HR | VOIS | Kolhapur |

| 108 | Swati | 30000 | employee | Cognizant | Mumbai |

| 109 | Ram | 30000 | employee | DXC | Mumbai |

| 110 | Piyush | 14000 | employee11 | TCS | Pune |

+-------+----------+--------+-------------+---------------+----------+

9 rows in set (0.01 sec)

**8) Find name of the employees whose salary is <15000 :**

mysql> select \* from employee where salary<15000;

+-------+----------+--------+-------------+---------------+------+

| empid | empname | salary | designation | company\_name | city |

+-------+----------+--------+-------------+---------------+------+

| 107 | Sara | 9000 | Supervisor | Tech Mahindra | Pune |

| 110 | Piyush | 14000 | employee11 | TCS | Pune |

+-------+----------+--------+-------------+---------------+------+

2 rows in set (0.00 sec)

**9) Update designation of employee 11 to ‘Asst. Prof.’ :**

mysql> update employee set designation="AsstProf" where designation="employee11";

Query OK, 1 row affected (0.11 sec)

Rows matched: 1 Changed: 1 Warnings: 0

mysql> select \* from employee where designation="AsstProf";

+-------+----------+--------+-------------+--------------+------+

| empid | empname | salary | designation | company\_name | city |

+-------+----------+--------+-------------+--------------+------+

| 110 | Piyush | 14000 | AsstProf | TCS | Pune |

+-------+----------+--------+-------------+--------------+------+

1 row in set (0.00 sec)

**10) Delete the employees having designation “supervisor” :**

mysql> delete from employee where designation="Supervisor";

Query OK, 1 row affected (0.22 sec)

mysql> select \* from employee where designation="Supervisor";

Empty set (0.00 sec)

**11) Increment the salary of employees by 5% :**

mysql> update employee set salary=salary+(salary\*5/100);

Query OK, 9 rows affected (0.17 sec)

Rows matched: 9 Changed: 9 Warnings: 0

mysql> select \* from employee;

+-------+----------+--------+-------------+---------------+----------+

| empid | empname | salary | designation | company\_name | city |

+-------+----------+--------+-------------+---------------+----------+

| 101 | Sakshi | 52500 | CEO | TCS | Pune |

| 102 | Chirag | 52500 | CEO | Tech Mahindra | Pune |

| 103 | Shivani | 42000 | Manager | KPIT | Mumbai |

| 104 | Ashu | 42000 | Manager | DXC | Kolhapur |

| 105 | Swara | 21000 | HR | DXC | Solapur |

| 106 | Swayam | 22050 | HR | VOIS | Kolhapur |

| 108 | Swati | 31500 | employee | Cognizant | Mumbai |

| 109 | Ram | 31500 | employee | DXC | Mumbai |

| 110 | Piyush | 14700 | AsstProf | TCS | Pune |

+-------+----------+--------+-------------+---------------+----------+

9 rows in set (0.00 sec)

**12) Find the name of employees having salary between 18000 and 22000 :**

mysql> select empname from employee where salary between 18000 and 22000;

+----------+

| empname |

+----------+

| Swara |

| Swati |

+----------+

2 rows in set (0.00 sec)

**13) Find number of employees working in each company :**

mysql> select company\_name, count(\*) from employee group by company\_name;

+---------------+----------+

| company\_name | count(\*) |

+---------------+----------+

| TCS | 2 |

| Tech Mahindra | 1 |

| KPIT | 1 |

| DXC | 3 |

| VOIS | 1 |

| Cognizant | 1 |

+---------------+----------+

6 rows in set (0.05 sec)

**14) Find employees having same salary as Ram :**

select \* from employee where salary IN(select salary from employee group by salary having count(salary)>1) and city="Mumbai";

+-------+----------+--------+-------------+--------------+--------+

| empid | empname | salary | designation | company\_name | city |

+-------+----------+--------+-------------+--------------+--------+

| 108 | Swati | 31500 | employee | Cognizant | Mumbai |

| 109 | Ram | 31500 | employee | DXC | Mumbai |

+-------+----------+--------+-------------+--------------+--------+

2 rows in set (0.00 sec)

**15) Find employee having maximum salary :**

mysql> select \* from employee where salary in (select max(salary)from employee);

+-------+----------+--------+-------------+---------------+------+

| empid | empname | salary | designation | company\_name | city |

+-------+----------+--------+-------------+---------------+------+

| 101 | Sakshi | 52500 | CEO | TCS | Pune |

| 102 | Chirag | 52500 | CEO | Tech Mahindra | Pune |

+-------+----------+--------+-------------+---------------+------+

2 rows in set (0.00 sec)

**16) Find company having highest average salary :**

mysql> select company\_name from employee where salary in(select max(salary)from employee);

+---------------+

| company\_name |

+---------------+

| TCS |

| Tech Mahindra |

+---------------+

2 rows in set (0.00 sec)

mysql> select company\_name from employee where salary>(select avg(salary)from

employee);

+---------------+

| company\_name |

+---------------+

| TCS |

| Tech Mahindra |

| KPIT |

| DXC |

+---------------+

4 rows in set (0.00 sec)

**Experiment No : 05**

**1. Even or Odd:**

SQL> connect system/manager;

Connected.

SQL> set serveroutput on;

SQL> declare

2 num1 number := &num1;

3 begin

4 if(mod(num1,2)=0)

5 then

6 dbms\_output.put\_line('num1 is even'||num1);

7 else

8 dbms\_output.put\_line('num1 is odd'||num1);

9 end if;

10 end;

11 /

Enter value for num1: 4

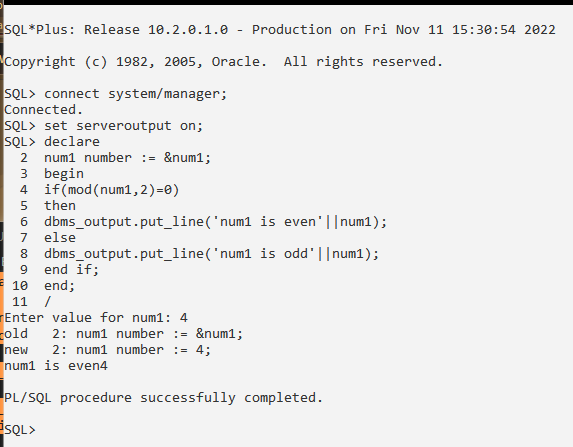
old 2: num1 number := &num1;

new 2: num1 number := 4;

num1 is even4

PL/SQL procedure successfully completed.

SQL>



**2.Stored Procedure :**

SQL\*Plus: Release 10.2.0.1.0 - Production on Fri Nov 11 15:30:54 2022

Copyright (c) 1982, 2005, Oracle. All rights reserved.

SQL> connect system/manager;

Connected.

SQL> set serveroutput on;

SQL> declare

2 num1 number := &num1;

3 begin

4 if(mod(num1,2)=0)

5 then

6 dbms\_output.put\_line('num1 is even'||num1);

7 else

8 dbms\_output.put\_line('num1 is odd'||num1);

9 end if;

10 end;

11 /

Enter value for num1: 4

old 2: num1 number := &num1;

new 2: num1 number := 4;

num1 is even4

PL/SQL procedure successfully completed.

SQL> create or replace procedure factorial2

2 (num1 in number)

3 is

4 fact1 number:=1;

5 begin

6 for i in 1..num1

7 loop

8 fact1:=fact1\*i;

9 end loop;

10 dbms\_output.put\_line('fact is:'||fact1);

11 end factorial2;

12 /

Procedure created.

SQL> declare

2 num1 number:=5;

3 begin

4 factorial2(num1);

5 end;

6 /

fact is:120

PL/SQL procedure successfully completed.

**3.Stored Function :**

SQL> create or replace function factorial

2 (num1 in number)

3 return number

4 is

5 fact1 number := 1;

6 begin

7 for i in 1..num1

8 loop

9 fact1 := fact1\*i;

10 end loop;

11 return fact1;

12 end factorial;

13 /

Function created.

SQL> declare

2 num1 number:= 6;

3 f number(5);

4 begin

5 f:= factorial(num1);

6 dbms\_output.put\_line('The factorial is :'||f);

7 end;

8 /

The factorial is :720

PL/SQL procedure successfully completed.

**4.Cursor :**

**Q. Write a PL/SQL cursor if salary of employee is greater than 10000 then increment it by 6 % otherwise by 5%.**

**Query:**

SQL\*Plus: Release 10.2.0.1.0 - Production on Tue Nov 15 11:52:02 2022

Copyright (c) 1982, 2005, Oracle. All rights reserved.

SQL> connect system/manager;

Connected.

SQL> set serveroutput on;

SQL> create table emp(eid int,ename varchar2(20),esalary number(5));

Table created.

SQL> insert into emp values(11,'Sakshi',30000);

1 row created.

SQL> insert into emp values(12,'Chirag',50000);

1 row created.

SQL> insert into emp values(13,'Rutika',20000);

1 row created.

SQL> insert into emp values(14,'Pallavi',9000);

1 row created.

SQL> insert into emp values(15,'Anamika',8000);

1 row created.

SQL> select \* from emp;

EID ENAME ESALARY

---------- -------------------- ----------

11 Sakshi 30000

12 Chirag 50000

13 Rutika 20000

14 Pallavi 9000

15 Anamika 8000

SQL> declare

2 empid emp.eid %type;

3 esal emp.esalary %type;

4 cursor s1 is select eid,esalary from emp;

5 begin

6 open s1;

7 loop

8 fetch s1 into empid,esal;

9 if(esal > 10000)

10 then

11 esal:= esal \* 1.06;

12 else

13 esal:= esal \* 1.05;

14 end if;

15 update emp set esalary=esal where eid=empid;

16 exit when s1 %notfound;

17 end loop;

18 close s1;

19 end;

20 /

PL/SQL procedure successfully completed.

SQL> select \* from emp;

EID ENAME ESALARY

---------- -------------------- ----------

11 Sakshi 31800

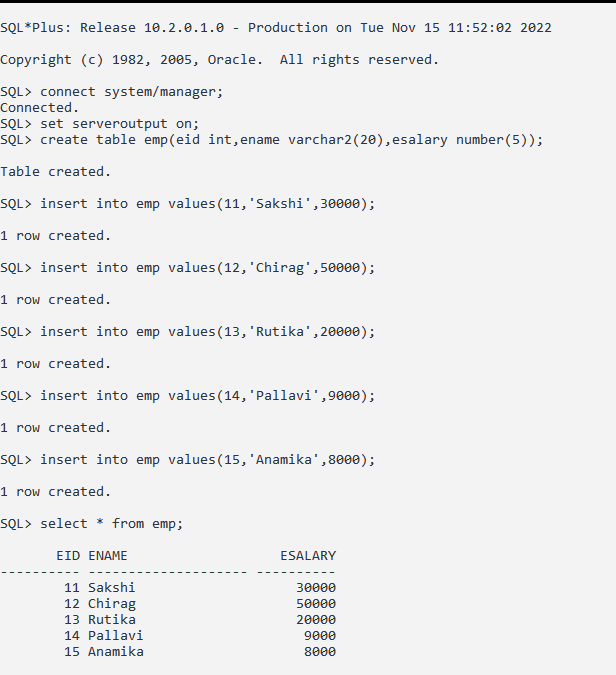
12 Chirag 53000

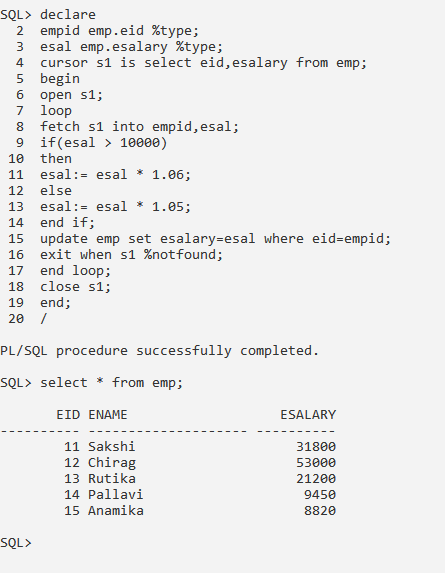
13 Rutika 21200

14 Pallavi 9450

15 Anamika 8820

SQL>





**Q. Write a cursor to calculate net Salary for above emp table.**

SQL> alter table emp add netsalary number(5);

Table altered.

SQL> alter table emp modify netsalary numeric(10,2);

Table altered.

SQL> declare

2 empid emp.eid %type;

3 esal emp.esalary %type;

4 nsal emp.netsalary %type;

5 cursor s1 is select eid,esalary from emp;

6 begin

7 open s1;

8 loop

9 fetch s1 into empid,esal;

10 nsal:=esal+(esal\*0.15)+(esal\*0.60)+(esal\*0.2);

11 update emp set netsalary = nsal where eid=empid;

12 exit when s1 %notfound;

13 end loop;

14 close s1;

15 end;

16 /

PL/SQL procedure successfully completed.

SQL> select \* from emp;

EID ENAME ESALARY NETSALARY

---------- -------------------- ---------- ----------

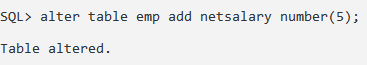
11 Sakshi 31800 62010

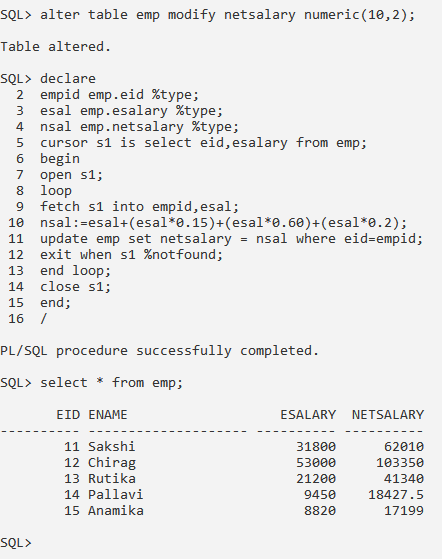
12 Chirag 53000 103350

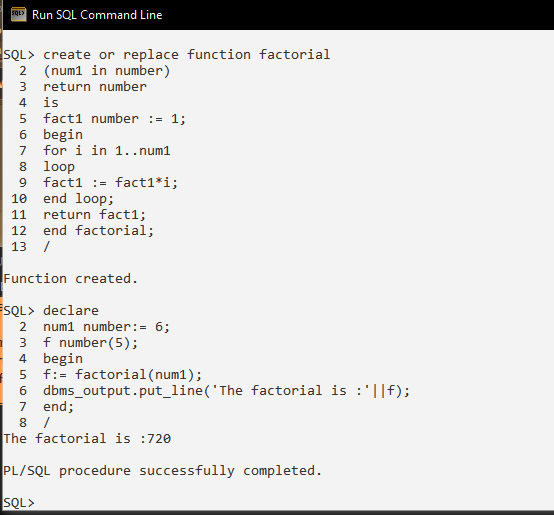
13 Rutika 21200 41340

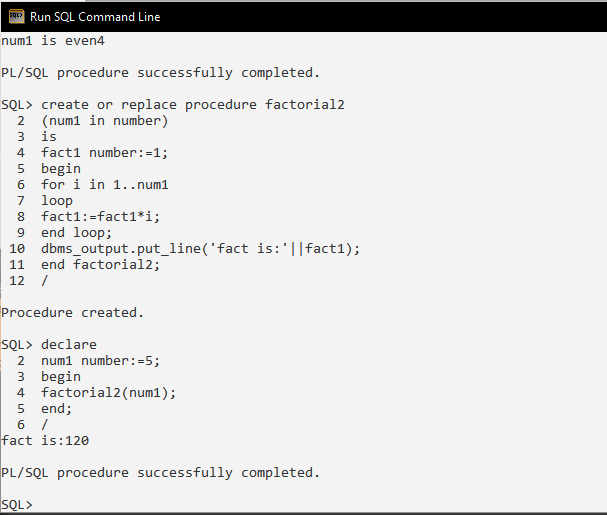
14 Pallavi 9450 18427.5

15 Anamika 8820 17199









**5.Trigger :**

**Trigger :**

**Example: Trigger before the insertion of the student record in the table:**

SQL\*Plus: Release 10.2.0.1.0 - Production on Tue Nov 15 12:45:38 2022

Copyright (c) 1982, 2005, Oracle. All rights reserved.

SQL> connect system/manager;

Connected.

SQL> set serveroutput on;

SQL> CREATE TABLE Student

2 (

3 RollNo INT NOT NULL PRIMARY KEY,

4 FirstName Varchar (20),

5 EnglishMarks number(5),

6 PhysicsMarks number(5),

7 ChemistryMarks number(5),

8 MathsMarks number(5),

9 TotalMarks number(5),

10 Percentage numeric(4,2)

11 );

Table created.

SQL>

SQL> CREATE or replace TRIGGER Student\_Marks1

2 BEFORE INSERT ON Student

3 FOR EACH ROW

4 begin

5 :new.TotalMarks := :new.EnglishMarks + :new.PhysicsMarks + :new.ChemistryMarks + :new.MathsMarks;

6 :new.Percentage := (:new.TotalMarks / 400) \* 100;

7 end;

8 /

Trigger created.

SQL> INSERT INTO Student (RollNo, FirstName, EnglishMarks, PhysicsMarks, ChemistryMarks, MathsMarks, TotalMarks, Percentage) VALUES ( 64, 'Sakshi', 88, 75, 69, 92, 0, 0);

1 row created.

SQL> INSERT INTO Student (RollNo, FirstName, EnglishMarks, PhysicsMarks, ChemistryMarks, MathsMarks, TotalMarks, Percentage) VALUES ( 78, 'Rutika', 88, 75, 60, 90, 0, 0);

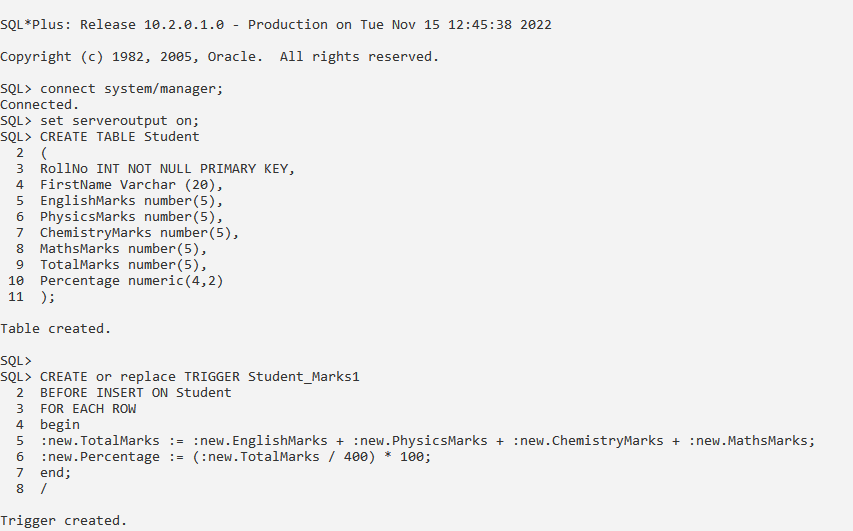
1 row created.

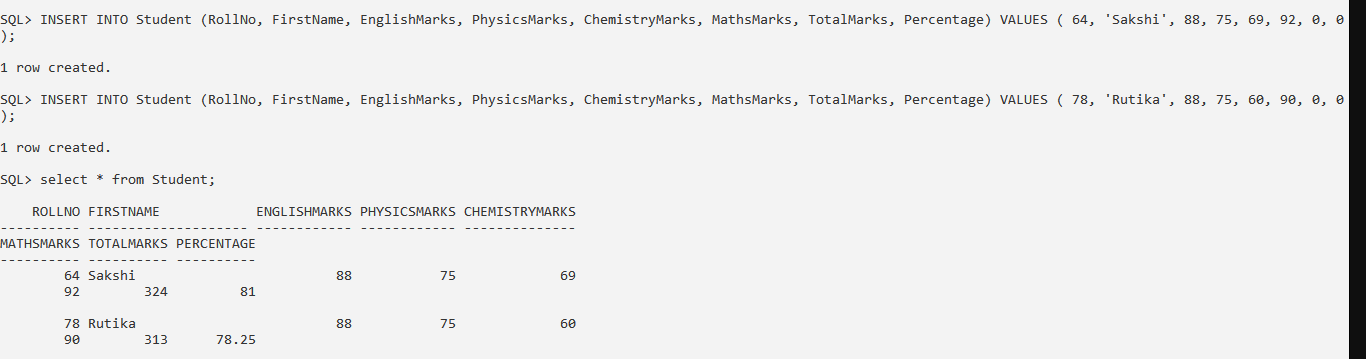
SQL> select \* from Student;

ROLLNO FIRSTNAME ENGLISHMARKS PHYSICSMARKS CHEMISTRYMARKS MATHSMARKS TOTALMARKS PERCENTAGE

64 Sakshi 88 75 69 92 324 81

78 Rutika 88 75 60 90 313 78.25





**Example 2 :** **Check the salary difference by procedure:**

SQL> create table customer (id number(5), name varchar(15), age number(5), address varchar(15), salary numeric(8,2));

Table created.

SQL> insert into customer values (1, 'Sakshi', 21, 'Kolhapur', 20000);

1 row created.

SQL> insert into customer values (2, 'Rutika', 22, 'Pune', 30000);

1 row created.

SQL> insert into customer values (3, 'Chirag', 23, 'Kolhapur', 25000);

1 row created.

SQL> select \* from customer;

ID NAME AGE ADDRESS SALARY

---------- --------------- ---------- --------------- ----------

1 Sakshi 21 Kolhapur 20000

2 Rutika 22 Pune 30000

3 Chirag 23 Kolhapur 25000

SQL> CREATE OR REPLACE TRIGGER display\_salary\_changes

2 BEFORE DELETE OR INSERT OR UPDATE ON customer

3 FOR EACH ROW

4 WHEN (NEW.id > 0)

5 DECLARE

6 sal\_diff number;

7 BEGIN

8 sal\_diff := :NEW.salary - :OLD.salary;

9 dbms\_output.put\_line('Old salary: ' || :OLD.salary);

10 dbms\_output.put\_line('New salary: ' || :NEW.salary);

11 dbms\_output.put\_line('Salary difference: ' || sal\_diff);

12 END;

13 /

Trigger created.

SQL> DECLARE

2 total\_rows number(2);

3 BEGIN

4 UPDATE customer

5 SET salary = salary + 5000;

6 IF sql%notfound THEN

7 dbms\_output.put\_line('no customers updated');

8 ELSIF sql%found THEN

9 total\_rows := sql%rowcount;

10 dbms\_output.put\_line( total\_rows || ' customers updated ');

11 END IF;

12 END;

13 /

Old salary: 20000

New salary: 25000

Salary difference: 5000

Old salary: 30000

New salary: 35000

Salary difference: 5000

Old salary: 25000

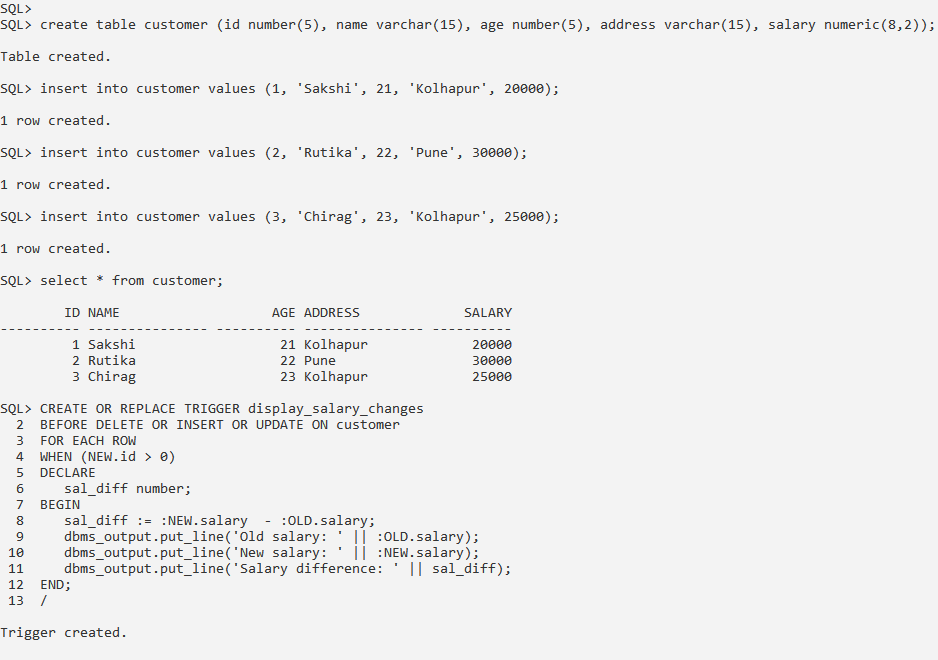
New salary: 30000

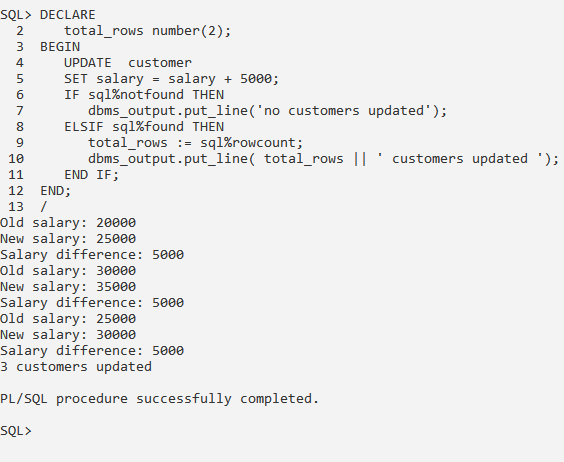
Salary difference: 5000

3 customers updated

PL/SQL procedure successfully completed.

SQL>





Assignment no. 7

MongoDB Queries

**Create db Employee**

> use cse switched to db cse

> db.createCollection("Employee")

{ "ok" : 1 }

> show collections

Employee

**Insert 10 employee documents**

>

db.Employee.insert({Firstname:"Ashutosh",Lastname:"Zende",gender:"F",salary:30000,deptname:" CSE"})

WriteResult({ "nInserted" : 1 })

>

db.Employee.insert({Firstname:"Anay",Lastname:"Zende",gender:"M",salary:25000,deptname:" CSE"})

WriteResult({ "nInserted" : 1 })

>

db.Employee.insert({Firstname:"Smita",Lastname:"Patil",gender:"F",salary:25000,deptname:" Mechanical"})

WriteResult({ "nInserted" : 1 })

>

db.Employee.insert({Firstname:"Tejas",Lastname:"Patil",gender:"M",salary:30000,deptname:"Rob otics"})

WriteResult({ "nInserted" : 1 })

>

db.Employee.insert({Firstname:"Abhishek",Lastname:"Mane",gender:"M",salary:35000,deptnam e:"ENTC"})

WriteResult({ "nInserted" : 1 })

>

db.Employee.insert({Firstname:"Ashutosh",Lastname:"Patil",gender:"M",salary:15000,deptname:" CSE"})

WriteResult({ "nInserted" : 1 })

>

db.Employee.insert({Firstname:"Sakshi",Lastname:"Bhosale",gender:"F",salary:35000,deptname:" CSE"})

WriteResult({ "nInserted" : 1 })

>

db.Employee.insert({Firstname:"Mahesh",Lastname:"Bhosale",gender:"M",salary:45000,deptname:" Mech"})

WriteResult({ "nInserted" : 1 })

>

db.Employee.insert({Firstname:" Prathmesh",Lastname:"Shinde",gender:"F",salary:45000,deptname:"M ech"})

WriteResult({ "nInserted" : 1 })>

db.Employee.insert({Firstname:"Girish",Lastname:"Gaikwad",gender:"F",salary:35000,deptnam e:"CSE"})

WriteResult({ "nInserted" : 1 })

> db.Employee.find()

{ "\_id" : ObjectId("638dde0d5b9cae48f3c90016"), "Firstname" : "Ashutosh", "Lastname" :

"Zende", "gender" : "F", "salary" : 30000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638dde535b9cae48f3c90017"), "Firstname" : "Anay", "Lastname" : "Zende",

"gender" : "M", "salary" : 25000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638ddef25b9cae48f3c9001a"), "Firstname" : "Smita", "Lastname" : "Patil",

"gender" : "F", "salary" : 25000, "deptname" : "Mechanical" }

{ "\_id" : ObjectId("638ddf405b9cae48f3c9001b"), "Firstname" : "Tejas", "Lastname" : "Patil",

"gender" : "M", "salary" : 30000, "deptname" : "Robotics" }

{ "\_id" : ObjectId("638de1655b9cae48f3c9001c"), "Firstname" : "Abhishek", "Lastname" :

"Mane", "gender" : "M", "salary" : 35000, "deptname" : "ENTC" }

{ "\_id" : ObjectId("638de1c05b9cae48f3c9001d"), "Firstname" : "Ashutosh", "Lastname" : "Patil",

"gender" : "M", "salary" : 15000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de1e75b9cae48f3c9001e"), "Firstname" : "Sakshi", "Lastname" : "Bhosale",

"gender" : "F", "salary" : 35000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de2245b9cae48f3c9001f"), "Firstname" : "Mahesh", "Lastname" : "Bhosale",

"gender" : "M", "salary" : 45000, "deptname" : "Mech" }

{ "\_id" : ObjectId("638de2685b9cae48f3c90021"), "Firstname" : " Prathmesh", "Lastname" : "Shinde",

"gender" : "F", "salary" : 45000, "deptname" : "Mech" }

{ "\_id" : ObjectId("638de29e5b9cae48f3c90022"), "Firstname" : "Girish", "Lastname" :

"Gaikwad", "gender" : "F", "salary" : 35000, "deptname" : "CSE" }

**Salary greater than 20000**

> db.Employee.find({salary:{$gt:20000}});

{ "\_id" : ObjectId("638dde0d5b9cae48f3c90016"), "Firstname" : "Ashutosh", "Lastname" :

"Zende", "gender" : "F", "salary" : 30000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638dde535b9cae48f3c90017"), "Firstname" : "Anay", "Lastname" : "Zende",

"gender" : "M", "salary" : 25000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638ddef25b9cae48f3c9001a"), "Firstname" : "Smita", "Lastname" : "Patil",

"gender" : "F", "salary" : 25000, "deptname" : "Mechanical" }

{ "\_id" : ObjectId("638ddf405b9cae48f3c9001b"), "Firstname" : "Tejas", "Lastname" : "Patil",

"gender" : "M", "salary" : 30000, "deptname" : "Robotics" }

{ "\_id" : ObjectId("638de1655b9cae48f3c9001c"), "Firstname" : "Abhishek", "Lastname" :

"Mane", "gender" : "M", "salary" : 35000, "deptname" : "ENTC" }

{ "\_id" : ObjectId("638de1e75b9cae48f3c9001e"), "Firstname" : "Sakshi", "Lastname" : "Bhosale",

"gender" : "F", "salary" : 35000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de2245b9cae48f3c9001f"), "Firstname" : "Mahesh", "Lastname" : "Bhosale",

"gender" : "M", "salary" : 45000, "deptname" : "Mech" }

{ "\_id" : ObjectId("638de2685b9cae48f3c90021"), "Firstname" : " Prathmesh", "Lastname" : "Shinde",

"gender" : "F", "salary" : 45000, "deptname" : "Mech" }

{ "\_id" : ObjectId("638de29e5b9cae48f3c90022"), "Firstname" : "Girish", "Lastname" : "Gaikwad", "gender" : "F", "salary" : 35000, "deptname" : "CSE" }

**Salary less than 20000**

> db.Employee.find({salary:{$lt:20000}});

{ "\_id" : ObjectId("638de1c05b9cae48f3c9001d"), "Firstname" : "Ashutosh", "Lastname" : "Patil", "gender" : "M", "salary" : 15000, "deptname" : "CSE" }

**Employee of CSE DEPT**

> db.Employee.find({deptname:'CSE'});

{ "\_id" : ObjectId("638dde0d5b9cae48f3c90016"), "Firstname" : "Ashutosh", "Lastname" :

"Zende", "gender" : "F", "salary" : 30000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638dde535b9cae48f3c90017"), "Firstname" : "Anay", "Lastname" : "Zende", "gender" : "M", "salary" : 25000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de1c05b9cae48f3c9001d"), "Firstname" : "Ashutosh", "Lastname" : "Patil",

"gender" : "M", "salary" : 15000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de1e75b9cae48f3c9001e"), "Firstname" : "Sakshi", "Lastname" : "Bhosale",

"gender" : "F", "salary" : 35000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de29e5b9cae48f3c90022"), "Firstname" : "Girish", "Lastname" : "Gaikwad", "gender" : "F", "salary" : 35000, "deptname" : "CSE" }

**Salary of Prathmesh**

> db.Employee.find({Firstname:' Prathmesh'},{salary:1});

{ "\_id" : ObjectId("638de2685b9cae48f3c90021"), "salary" : 45000 }

**Max salary of dept**

> db.Employee.aggregate([{$group:{\_id:'$deptname',Maximumsalary:{$max:'$salary'}}}]);

{ "\_id" : "Mech", "Maximumsalary" : 45000 }

{ "\_id" : "CSE", "Maximumsalary" : 35000 }

{ "\_id" : "Robotics", "Maximumsalary" : 30000 }

{ "\_id" : "ENTC", "Maximumsalary" : 35000 }

{ "\_id" : "Mechanical", "Maximumsalary" : 25000 }

**Update Salary by 10%**

> db.Employee.update({},{$mul:{salary:1.1}},{multi:true});

WriteResult({ "nMatched" : 10, "nUpserted" : 0, "nModified" : 10 })

> db.Employee.find();

{ "\_id" : ObjectId("638dde0d5b9cae48f3c90016"), "Firstname" : "Ashutosh", "Lastname" :

"Zende", "gender" : "F", "salary" : 33000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638dde535b9cae48f3c90017"), "Firstname" : "Anay", "Lastname" : "Zende",

"gender" : "M", "salary" : 27500.000000000004, "deptname" : "CSE" }

{ "\_id" : ObjectId("638ddef25b9cae48f3c9001a"), "Firstname" : "Smita", "Lastname" : "Patil",

"gender" : "F", "salary" : 27500.000000000004, "deptname" : "Mechanical" }

{ "\_id" : ObjectId("638ddf405b9cae48f3c9001b"), "Firstname" : "Tejas", "Lastname" : "Patil",

"gender" : "M", "salary" : 33000, "deptname" : "Robotics" }

{ "\_id" : ObjectId("638de1655b9cae48f3c9001c"), "Firstname" : "Abhishek", "Lastname" :

"Mane", "gender" : "M", "salary" : 38500, "deptname" : "ENTC" }

{ "\_id" : ObjectId("638de1c05b9cae48f3c9001d"), "Firstname" : "Ashutosh", "Lastname" : "Patil",

"gender" : "M", "salary" : 16500, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de1e75b9cae48f3c9001e"), "Firstname" : "Sakshi", "Lastname" : "Bhosale",

"gender" : "F", "salary" : 38500, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de2245b9cae48f3c9001f"), "Firstname" : "Mahesh", "Lastname" : "Bhosale",

"gender" : "M", "salary" : 49500.00000000001, "deptname" : "Mech" }

{ "\_id" : ObjectId("638de2685b9cae48f3c90021"), "Firstname" : " Prathmesh", "Lastname" : "Shinde",

"gender" : "F", "salary" : 49500.00000000001, "deptname" : "Mech" }

{ "\_id" : ObjectId("638de29e5b9cae48f3c90022"), "Firstname" : "Girish", "Lastname" : "Gaikwad", "gender" : "F", "salary" : 38500, "deptname" : "CSE" }

**Update deptname of Prathmesh to CSE**

> db.Employee.update({Firstname:" Prathmesh"},{$set:{deptname:'CSE'}});

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.Employee.find();

{ "\_id" : ObjectId("638dde0d5b9cae48f3c90016"), "Firstname" : "Ashutosh", "Lastname" :

"Zende", "gender" : "F", "salary" : 33000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638dde535b9cae48f3c90017"), "Firstname" : "Anay", "Lastname" : "Zende",

"gender" : "M", "salary" : 27500.000000000004, "deptname" : "CSE" }

{ "\_id" : ObjectId("638ddef25b9cae48f3c9001a"), "Firstname" : "Smita", "Lastname" : "Patil",

"gender" : "F", "salary" : 27500.000000000004, "deptname" : "Mechanical" }

{ "\_id" : ObjectId("638ddf405b9cae48f3c9001b"), "Firstname" : "Tejas", "Lastname" : "Patil",

"gender" : "M", "salary" : 33000, "deptname" : "Robotics" }

{ "\_id" : ObjectId("638de1655b9cae48f3c9001c"), "Firstname" : "Abhishek", "Lastname" :

"Mane", "gender" : "M", "salary" : 38500, "deptname" : "ENTC" }

{ "\_id" : ObjectId("638de1c05b9cae48f3c9001d"), "Firstname" : "Ashutosh", "Lastname" : "Patil",

"gender" : "M", "salary" : 16500, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de1e75b9cae48f3c9001e"), "Firstname" : "Sakshi", "Lastname" : "Bhosale",

"gender" : "F", "salary" : 38500, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de2245b9cae48f3c9001f"), "Firstname" : "Mahesh", "Lastname" : "Bhosale",

"gender" : "M", "salary" : 49500.00000000001, "deptname" : "Mech" }

{ "\_id" : ObjectId("638de2685b9cae48f3c90021"), "Firstname" : " Prathmesh", "Lastname" : "Shinde",

"gender" : "F", "salary" : 49500.00000000001, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de29e5b9cae48f3c90022"), "Firstname" : "Girish", "Lastname" : "Gaikwad", "gender" : "F", "salary" : 38500, "deptname" : "CSE" }

**Display employees salary in descending order**

> db.Employee.find({},{Firstname:1,salary:1,\_id:0}).sort({salary:-1});

{ "Firstname" : "Mahesh", "salary" : 49500.00000000001 }

{ "Firstname" : " Prathmesh", "salary" : 49500.00000000001 }

{ "Firstname" : "Abhishek", "salary" : 38500 }

{ "Firstname" : "Sakshi", "salary" : 38500 }

{ "Firstname" : "Girish", "salary" : 38500 }

{ "Firstname" : "Ashutosh", "salary" : 33000 }

{ "Firstname" : "Tejas", "salary" : 33000 }

{ "Firstname" : "Anay", "salary" : 27500.000000000004 }

{ "Firstname" : "Smita", "salary" : 27500.000000000004 }

{ "Firstname" : "Ashutosh", "salary" : 16500 }

**Find employee whose salary is 33000**

> db.Employee.find({salary:{$eq:33000}});

{ "\_id" : ObjectId("638dde0d5b9cae48f3c90016"), "Firstname" : "Ashutosh", "Lastname" :

"Zende", "gender" : "F", "salary" : 33000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638ddf405b9cae48f3c9001b"), "Firstname" : "Tejas", "Lastname" : "Patil", "gender" : "M", "salary" : 33000, "deptname" : "Robotics" }

**Find top 5 employees having highest salary**

> db.Employee.find({},{Firstname:1,salary:1,\_id:0}).sort({salary:-1}).limit(5);

{ "Firstname" : " Prathmesh", "salary" : 49500.00000000001 }

{ "Firstname" : "Mahesh", "salary" : 49500.00000000001 }

{ "Firstname" : "Girish", "salary" : 38500 }

{ "Firstname" : "Abhishek", "salary" : 38500 }

{ "Firstname" : "Sakshi", "salary" : 38500 }

**Delete document of Anay**

> db.Employee.remove({Firstname:"Anay"})

WriteResult({ "nRemoved" : 1 })

> db.Employee.find()

{ "\_id" : ObjectId("638dde0d5b9cae48f3c90016"), "Firstname" : "Ashutosh", "Lastname" :

"Zende", "gender" : "F", "salary" : 33000, "deptname" : "CSE" }

{ "\_id" : ObjectId("638ddef25b9cae48f3c9001a"), "Firstname" : "Smita", "Lastname" : "Patil",

"gender" : "F", "salary" : 27500.000000000004, "deptname" : "Mechanical" }

{ "\_id" : ObjectId("638ddf405b9cae48f3c9001b"), "Firstname" : "Tejas", "Lastname" : "Patil",

"gender" : "M", "salary" : 33000, "deptname" : "Robotics" }

{ "\_id" : ObjectId("638de1655b9cae48f3c9001c"), "Firstname" : "Abhishek", "Lastname" :

"Mane", "gender" : "M", "salary" : 38500, "deptname" : "ENTC" }

{ "\_id" : ObjectId("638de1c05b9cae48f3c9001d"), "Firstname" : "Ashutosh", "Lastname" : "Patil",

"gender" : "M", "salary" : 16500, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de1e75b9cae48f3c9001e"), "Firstname" : "Sakshi", "Lastname" : "Bhosale",

"gender" : "F", "salary" : 38500, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de2245b9cae48f3c9001f"), "Firstname" : "Mahesh", "Lastname" : "Bhosale",

"gender" : "M", "salary" : 49500.00000000001, "deptname" : "Mech" }

{ "\_id" : ObjectId("638de2685b9cae48f3c90021"), "Firstname" : " Prathmesh", "Lastname" : "Shinde",

"gender" : "F", "salary" : 49500.00000000001, "deptname" : "CSE" }

{ "\_id" : ObjectId("638de29e5b9cae48f3c90022"), "Firstname" : "Girish", "Lastname" : "Gaikwad", "gender" : "F", "salary" : 38500, "deptname" : "CSE" } >

**http://beginner-sql-tutorial.com/sql-grant-revoke-privileges-roles.htm**

**SQL GRANT REVOKE Commands**

DCL commands are used to enforce database security in a multiple user database environment. Two types of DCL commands are GRANT and REVOKE. Only Database Administrator's or owner's of the database object can provide/remove privileges on a database object.

**SQL GRANT Command**

SQL GRANT is a command used to provide access or privileges on the database objects to the users.

**The Syntax for the GRANT command is:**

GRANT privilege\_name   
ON object\_name   
TO {user\_name |PUBLIC |role\_name}   
[WITH GRANT OPTION];

* ***privilege\_name*** is the access right or privilege granted to the user. Some of the access rights are ALL, EXECUTE, and SELECT.
* ***object\_name*** is the name of an database object like TABLE, VIEW, STORED PROC and SEQUENCE.
* ***user\_name*** is the name of the user to whom an access right is being granted.
* ***user\_name*** is the name of the user to whom an access right is being granted.
* ***PUBLIC*** is used to grant access rights to all users.
* ***ROLES*** are a set of privileges grouped together.
* ***WITH GRANT OPTION*** - allows a user to grant access rights to other users.

**For Example:** GRANT SELECT ON employee TO user1; This command grants a SELECT permission on employee table to user1.You should use the WITH GRANT option carefully because for example if you GRANT SELECT privilege on employee table to user1 using the WITH GRANT option, then user1 can GRANT SELECT privilege on employee table to another user, such as user2 etc. Later, if you REVOKE the SELECT privilege on employee from user1, still user2 will have SELECT privilege on employee table.

**SQL REVOKE Command:**

The REVOKE command removes user access rights or privileges to the database objects.

The Syntax for the REVOKE command is:

REVOKE privilege\_name   
ON object\_name   
FROM {user\_name |PUBLIC |role\_name}

**For Example:** REVOKE SELECT ON employee FROM user1;This command will REVOKE a SELECT privilege on employee table from user1.When you REVOKE SELECT privilege on a table from a user, the user will not be able to SELECT data from that table anymore. However, if the user has received SELECT privileges on that table from more than one users, he/she can SELECT from that table until everyone who granted the permission revokes it. You cannot REVOKE privileges if they were not initially granted by you.

**Privileges and Roles:**

Privileges: Privileges defines the access rights provided to a user on a database object. There are two types of privileges.

1. **System privileges** - This allows the user to CREATE, ALTER, or DROP database objects.   
   **2) Object privileges** - This allows the user to EXECUTE, SELECT, INSERT, UPDATE, or DELETE data from database objects to which the privileges apply.

Few CREATE system privileges are listed below:

|  |  |
| --- | --- |
| **System Privileges** | **Description** |
| CREATE object | allows users to create the specified object in their own schema. |
| CREATE ANY object | allows users to create the specified object in any schema. |

**The above rules also apply for ALTER and DROP system privileges.**

Few of the object privileges are listed below:

|  |  |
| --- | --- |
| **Object Privileges** | **Description** |
| INSERT | allows users to insert rows into a table. |
| SELECT | allows users to select data from a database object. |
| UPDATE | allows user to update data in a table. |
| EXECUTE | allows user to execute a stored procedure or a function. |

**Roles:**Roles are a collection of privileges or access rights. When there are many users in a database it becomes difficult to grant or revoke privileges to users. Therefore, if you define roles, you can grant or revoke privileges to users, thereby automatically granting or revoking privileges. You can either create Roles or use the system roles pre-defined by oracle.

Some of the privileges granted to the system roles are as given below:

|  |  |
| --- | --- |
| **System Role** | **Privileges Granted to the Role** |
| CONNECT | CREATE TABLE, CREATE VIEW, CREATE SYNONYM, CREATE SEQUENCE, CREATE SESSION etc. |
| RESOURCE | CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER etc. The primary usage of the RESOURCE role is to restrict access to database objects. |
| DBA | ALL SYSTEM PRIVILEGES |

**Creating Roles:**

**The Syntax to create a role is:**

CREATE ROLE role\_name   
[IDENTIFIED BY password];

**For Example:** To create a role called "developer" with password as "pwd", the code will be as follows

CREATE ROLE testing   
[IDENTIFIED BY pwd];

It's easier to GRANT or REVOKE privileges to the users through a role rather than assigning a privilege directly to every user. If a role is identified by a password, then, when you GRANT or REVOKE privileges to the role, you definitely have to identify it with the password.

We can GRANT or REVOKE privilege to a role as below.

**For example:** To grant CREATE TABLE privilege to a user by creating a testing role:

First, create a testing Role

CREATE ROLE testing

Second, grant a CREATE TABLE privilege to the ROLE testing. You can add more privileges to the ROLE.

GRANT CREATE TABLE TO testing;

Third, grant the role to a user.

GRANT testing TO user1;

To revoke a CREATE TABLE privilege from testing ROLE, you can write:

REVOKE CREATE TABLE FROM testing;

**The Syntax to drop a role from the database is as below:**

DROP ROLE role\_name;

**For example:** To drop a role called developer, you can write:

DROP ROLE testing;

***REVOKE Statement Syntax***

REVOKE [ GRANT OPTION FOR ] *privilege\_list*

ON *object*

FROM *user\_name* [ RESTRICT | CASCADE ]

The REVOKE statement takes privileges away from users. The arguments are similar to the GRANT statement. The major difference is the additional RESTRICT or CASCADE keyword and the GRANT OPTION FOR clause. The following describes the optional clauses GRANT OPTION FOR and RESTRICT or CASCADE.

NOTE: If none of the privileges that you are trying to revoke actually exist, an error is raised.

*RESTRICT | CASCADE*

If you use RESTRICT keyword, the privilege will be revoked only from the specified user. If the specified user granted had the WITH GRANT OPTION and granted the same privilege to other users, they will retain the privilege.

If you use CASCADE, it will revoke the privilege and any dependent privileges as a result of your grant. A dependent privilege is one that could exist, if you granted the privilege that you're trying to revoke, which is what you are trying to achieve as a result of your REVOKE statement.

If the optional RESTRICT or CASCADE keywords are not used, PointBase uses RESTRICT by default.

*GRANT OPTION FOR*

If he optional GRANT OPTION FOR clause is used, the WITH GRANT OPTION right is revoked. The actual privilege itself is not revoked. the GRANT OPTION is revoked. CASCADE and RESTRICT may be used in the same way as a normal REVOKE statement.

By the time we finish setting up a Catalog and granting appropriate Privileges to our users, we probably have several thousand Privilege descriptors in INFORMATION\_SCHEMA — more than the count for all other Schema Objects combined. Maintaining them is made easier by the fact that when an Object is dropped, the DBMS will silently destroy all associated Privilege descriptors. That leaves only the problem: how do we adjust for the occasional necessity to remove a Privilege descriptor due to a change in status of a particular User (or Role)? The problem does not occur frequently, but can be mightily cumbersome: the SQL Standard devotes about 40 pages to it. We have managed to simplify the description somewhat, by focussing on the two "essentials" of the process:

1. What we are trying to do is reverse the effects of a GRANT statement, using a REVOKE statement — the clauses of which have almost the same syntax as GRANT's clauses.
2. What we are really doing is deleting Privilege descriptor rows from INFORMATION\_SCHEMA.

The REVOKE statement destroys both Privilege descriptors and Role authorizations and so has two different syntaxes. The first is identified as the <revoke privilege statement> and the second as the <revoke role statement>. The required syntax for the REVOKE statement is:

<revoke privilege statement> ::=REVOKE [ GRANT OPTION FOR ] <privileges> FROM <grantee> [ {,<grantee>}... ]

[ FROM {CURRENT\_USER | CURRENT\_ROLE} ] {RESTRICT | CASCADE}

<revoke role statement> ::=

REVOKE [ ADMIN OPTION FOR ] <Role name> [ {,<Role name>}... ]

FROM <grantee> [ {,<grantee>}... ]

[ FROM {CURRENT\_USER | CURRENT\_ROLE} ]

{RESTRICT | CASCADE}

<grantee> ::= PUBLIC | <AuthorizationID>

The <revoke privilege statement> revokes one or more Privileges on a given Object from one or more grantees, including (possibly) PUBLIC, while the <revoke role statement> revokes the use of one or more Roles from one or more grantees. Only the grantor of the Privileges (or the Roles) may revoke them.

We've already shown you the syntax for the <privileges> Privilege specification; it's used exactly that way in the <revoke privilege statement> form of the REVOKE statement. Here are some examples:

**REVOKE** **SELECT** **ON** **TABLE** Table\_1 **FROM** **PUBLIC** **CASCADE**;

**REVOKE** **INSERT**(column\_1,column\_5) **ON** Table\_1 **FROM** sam **CASCADE**;

**REVOKE** **ALL** **PRIVILEGES** **ON** **TABLE** Table\_1 **FROM** **PUBLIC** **CASCADE**;

**REVOKE** **USAGE** **ON** **DOMAIN** domain\_1 **FROM** bob **CASCADE**;

**REVOKE** **EXECUTE** **ON** **SPECIFIC** **ROUTINE** some\_routine **FROM** sam **CASCADE**;

And here's an example of the <revoke role statement> form of REVOKE:

**REVOKE** assistants\_role **FROM** **PUBLIC** **CASCADE**;

In both cases, if your <grantee> is PUBLIC, you're revoking the Privilege (or the use of the Role) from a list of <grantee>s that contains all of the <AuthorizationID>s in the SQL-environment. If your <grantee> is one or more <AuthorizationID>s, you're revoking the Privilege (or the use of the Role) only from those <AuthorizationID>s. (Remember that an <AuthorizationID> may identify either a User or a Role.)

Remember that, for Tables, GRANT creates Privilege descriptors for both the Table and its Columns. Well, when youREVOKE a Table Privilege, all by-Column Privileges for that Table disappear. The effect is a bit strange; when you revoke a Table Privilege, you lose the Column Privilege (even if it was granted separately) and when you revoke a Column Privilege, you lose that Column Privilege — even if it resulted from a Table Privilege GRANT.

The optional FROM clause names the grantor of the Privileges or the Role you're revoking; CURRENT\_USER is the <AuthorizationID> of the current user and CURRENT\_ROLE is the <AuthorizationID> of the current Role. If you omit the clause, it defaults to FROM CURRENT\_USER — but if CURRENT\_USER is NULL, the clause defaults to FROM CURRENT\_ROLE. If you specify FROM CURRENT\_USER and the current <AuthorizationID> is a <Role name>, or if you specifyFROM CURRENT\_ROLE and the current <Role name> is NULL, the REVOKE statement will fail: your DBMS will return theSQLSTATE error 0L000 "invalid grantor". Here are two examples:

**REVOKE** **UPDATE** **ON** Table\_1 **FROM** sam **FROM** **CURRENT\_USER** **CASCADE**;

-- revokes the UPDATE Privilege on TABLE\_1 from Sam only if the current user

-- granted that Privilege in the first place

**REVOKE** assistants\_role **FROM** **PUBLIC** **FROM** **CURRENT\_ROLE** **CASCADE**;

-- revokes the use of the ASSISTANTS\_ROLE Role from PUBLIC only if the current

-- Role granted the use of that Role in the first place

The optional HIERARCHY OPTION FOR clause (applicable only to the <revoke privilege statement>) allows you to revoke only the WITH HIERARCHY OPTION from the specified SELECT Privilege.

The optional GRANT OPTION FOR (<revoke privilege statement>) and ADMIN OPTION FOR (<revoke role statement>) clauses allow you to revoke only the grantability of a Privilege or a Role. For example, consider these SQL statements:

**GRANT** **UPDATE** **ON** **TABLE** Table\_1 **TO** sam **WITH** **GRANT** **OPTION**;

**REVOKE** **GRANT** **OPTION** **FOR** **UPDATE** **ON** Table\_1 **FROM** sam **CASCADE**;

The first SQL statement allows Sam to update Table\_1, and to pass this Privilege on to others. The second SQL statement revokes the latter ability: Sam can still update Table\_1, but may no longer pass the Privilege on. Here's another example:

**GRANT** assistants\_role **TO** bob **WITH** **ADMIN** **OPTION**;

**REVOKE** **WITH** **ADMIN** **OPTION** **FOR** assistants\_role **FROM** bob **CASCADE**;

The first SQL statement allows Bob to use all of the Privileges belonging to the assistants\_role Role, and to pass the use of this Role on to others. The second SQL statement revokes the latter ability: Bob can still use the Role's Privileges, but may no longer pass that use on.

The GRANT/ADMIN option clauses have another effect. Suppose that a user holds a Privilege on a TableWITH GRANT OPTION, and does so, also with GRANT OPTION. The second user can now do the same for a third user, and so on — for example:

**GRANT** **DELETE** **ON** **TABLE** Sally\_Dates **TO** joe **WITH** **GRANT** **OPTION**;

-- assume Sally does this

**GRANT** **DELETE** **ON** **TABLE** Sally\_Dates **TO** sam **WITH** **GRANT** **OPTION**;

-- assume Joe does this

**GRANT** **DELETE** **ON** **TABLE** Sally\_Dates **TO** bob **WITH** **GRANT** **OPTION**;

-- assume Sam does this

What should happen if Sally now does:

**REVOKE** **DELETE** **ON** **TABLE** Sally\_Dates **FROM** joe;

Here, we've deliberately left off RESTRICT/CASCADE for the sake of the example, so let's assume that the SQL statement works: Joe no longer has the DELETE Privilege on Sally\_Dates. The Privileges handed down from Joe to Sam, and from Sam to Bob, are now called "abandoned Privileges": they are dependent on Joe's DELETE Privilege — and Joe doesn't have it any more. This is where the RESTRICT/CASCADE <keyword>s come in.

* If your REVOKE statement specifies CASCADE, the REVOKE succeeds — and it cascades down to revoke any Privileges that would otherwise be abandoned. In our example, that means both Sam and Bob would no longer have the DELETE Privilege on Sally\_Dates either.
* If your REVOKE statement specifies RESTRICT, the REVOKE succeeds only if the Privilege being revoked has no dependent Privileges. In our example, that means the REVOKE statement would fail.

The same holds true for revoking the use of a Role.

Objects can also become "abandoned" when a Privilege or the use of a Role is revoked. For example, remember that Joe holds the SELECT Privilege on Sally\_Dates and, with this, was able to create his View, JOE\_VIEWS. Now suppose Sally does this:

**REVOKE** **SELECT** **ON** Sally\_Dates **FROM** joe **CASCADE**;

The effect is that, not only does Joe lose his ability to SELECT from Sally\_Dates, but that JOE\_VIEWS is dropped! The reason is that, in effect, JOE\_VIEWS is nothing but a SELECT that Joe does from Sally\_Dates, and since such SELECTs are no longer allowed, the View may no longer exist. If, on the other hand, Sally does the following:

**REVOKE** **SELECT** **ON** Sally\_Dates **FROM** joe **RESTRICT**;

the effect is that the REVOKE statement fails: Sally may not revoke Joe's ability to SELECT from Sally\_Dates because this would mean that JOE\_VIEWS would be abandoned — and this is not allowed. The same holds true for any Object that anyone was able to create only because they held some required Privilege (or were able to use a Role that held that Privilege): if REVOKE ... RESTRICT is used, the statement will fail but if REVOKE ... CASCADE is used, the statement will not only revoke but drop all Objects that would otherwise be abandoned.

If the REVOKE statement isn't able to find a Privilege descriptor for every one of its Privilege specifications, your DBMS will return the SQLSTATE warning 01006 "warning-privilege not revoked".

If you want to restrict your code to Core SQL, don't use the <revoke role statement> form of the REVOKE statement and don't use REVOKE ... CASCADE or the GRANT OPTION FOR or HIERARCHY OPTION FOR clauses. Also, when revoking, make sure that your current <AuthorizationID> is the owner of the Schema that owns the Object you're revoking Privileges for.

**Vertical fragmentation**

* **Create Database :**

mysql> show databases;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| B.Tech\_37 |

| BE\_A\_37 |

| BE\_Harsha |

| BTech\_78 |

| Bank |

| b1\_4 |

| btech39 |

| db\_name |

| horizontal |

| mysql |

| performance\_schema |

| server\_DB |

| sm82 |

| sys |

| vertical |

+--------------------+

16 rows in set (0.12 sec)

mysql> use horizontal;

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Database changed

* **Create Table :**

create table employee(eid int,ename varchar(20),esalary int,ecity varchar(20));

insert into employee values(101,Sakshi,10000,Kolhapur);

insert into employee values(102,Rutika,20000,Solapur);

insert into employee values(103,Pranoti,30000,Pune);

insert into employee values(104,Aarya,40000,Kolhapur);

insert into employee values(105,Teju,50000,Mumbai);

select \* from employee;

mysql> select \* from employee;

+------+---------+---------+----------+

| eid | ename | esalary | ecity |

+------+---------+---------+----------+

| 101 | Sakshi | 10000 | kolhapur |

| 102 | Rutika | 20000 | solapur |

| 103 | Pranoti | 30000 | pune |

| 104 | Aarya | 40000 | kolhapur |

| 105 | Teju | 50000 | mumbai |

+------+---------+---------+----------+

5 rows in set (0.03 sec)

create table empkop(eid int,ename varchar(20),esalary int,ecity varchar(20));

mysql> select \* from empkop;

Empty set (0.00 sec)

**Server.java**

import java.util.\*;

import java.sql.\*;

import java.net.\*;

import java.io.\*;

public class hfs

{

public static void main(String args[])

{

DataInputStream dis;

DataOutputStream dos;

InputStream is;

OutputStream os;

Scanner s1=new Scanner(System.in);

try

{

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/horizontal?useSSL=false","root","root");

Statement st=con.createStatement();

ServerSocket s=new ServerSocket(1520);

Socket s2=s.accept();

is=s2.getInputStream();

os=s2.getOutputStream();

dis=new DataInputStream(is);

dos=new DataOutputStream(os);

int count=0;

String rw=new String(dis.readUTF());

System.out.println(rw);

ResultSet rs1=st.executeQuery(rw);

while(rs1.next())

{

int id=rs1.getInt(1);

String nm=rs1.getString(2);

int sal=rs1.getInt(3);

String ect=rs1.getString(4);

System.out.println(id+"\t"+nm+"\t"+sal+"\t"+ect);

count++;

}

dos.writeInt(count);

String nm;

int id,sal;

String ect;

ResultSet rs2=st.executeQuery(rw);

while(rs2.next())

{

id=rs2.getInt(1); nm=rs2.getString(2);

sal=rs2.getInt(3); ect=rs2.getString(4);

dos.writeInt(id); dos.writeUTF(nm);

dos.writeInt(sal); dos.writeUTF(ect);

}

}

catch(Exception e)

{

System.out.println(e);

}

} }

**Client.java**

import java.util.\*;

import java.sql.\*;

import java.net.\*;

import java.io.\*;

public class hfc

{

public static void main(String args[])

{

DataInputStream dis; DataOutputStream dos;

InputStream is; OutputStream os;

Statement st,st1,st2,st3;

Scanner s1=new Scanner(System.in);

try {

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/horizontal?useSSL=false","root","root");

st=con.createStatement();

Socket s=new Socket("127.0.0.1",1520);

is=s.getInputStream();

os=s.getOutputStream();

dis=new DataInputStream(is);

dos=new DataOutputStream(os);

System.out.println("Enter the query");

String rw=s1.nextLine();

dos.writeUTF(rw);

System.out.println("Horizontal fragmentation");

int id,sal; String nm; String ect;

int count=dis.readInt();

for(int i=0;i<count;i++)

{

id= dis.readInt();

nm=dis.readUTF();

sal= dis.readInt();

ect=dis.readUTF();

st.executeUpdate("insert into empkop values("+id+",'"+nm+"',"+sal+",'"+ect+"')");

}

ResultSet rs1=st.executeQuery("select \* from empkop ");

while(rs1.next())

{ int id1=rs1.getInt(1); nm=rs1.getString(2);

sal=rs1.getInt(3); ect=rs1.getString(4);

System.out.println(id1+"\t"+nm+"\t"+sal+"\t"+ect);

}

}

catch(Exception e)

{ System.out.println(e); }

} }

**output:**

server.java

ubuntu@ubuntu-OptiPlex-3010:~/Desktop/BTech\_78/horizontal$ javac hfs.java

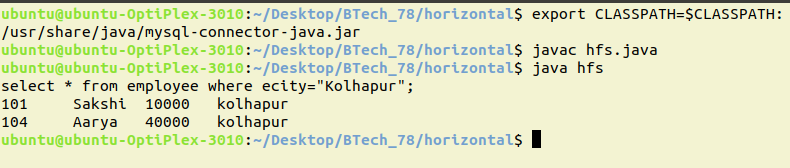
ubuntu@ubuntu-OptiPlex-3010:~/Desktop/BTech\_78/horizontal$ java hfs

select \* from employee where ecity="Kolhapur";

101 Sakshi 10000 kolhapur

104 Aarya 40000 kolhapur

ubuntu@ubuntu-OptiPlex-3010:~/Desktop/BTech\_78/horizontal$



client.java

ubuntu@ubuntu-OptiPlex-3010:~/Desktop/BTech\_78/horizontal$ export CLASSPATH=$CLASSPATH:/usr/share/java/mysql-connector-java.jar

ubuntu@ubuntu-OptiPlex-3010:~/Desktop/BTech\_78/horizontal$ javac hfc.java

ubuntu@ubuntu-OptiPlex-3010:~/Desktop/BTech\_78/horizontal$ java hfc

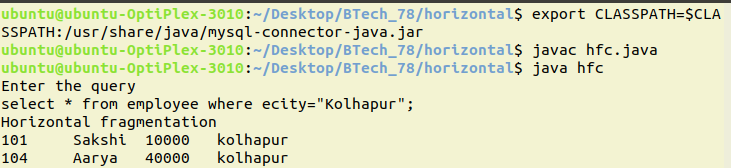
Enter the query

select \* from employee where ecity="Kolhapur";

Horizontal fragmentation

101 Sakshi 10000 kolhapur

104 Aarya 40000 kolhapur



mysql :

mysql> select \* from employee;

+------+---------+---------+----------+

| eid | ename | esalary | ecity |

+------+---------+---------+----------+

| 101 | Sakshi | 10000 | kolhapur |

| 102 | Rutika | 20000 | solapur |

| 103 | Pranoti | 30000 | pune |

| 104 | Aarya | 40000 | kolhapur |

| 105 | Teju | 50000 | mumbai |

+------+---------+---------+----------+

5 rows in set (0.00 sec)

mysql> select \* from empkop;

+------+--------+---------+----------+

| eid | ename | esalary | ecity |

+------+--------+---------+----------+

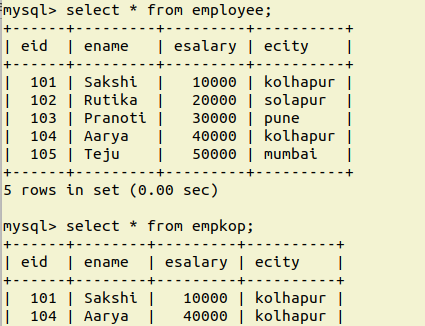
| 101 | Sakshi | 10000 | kolhapur |

| 104 | Aarya | 40000 | kolhapur |

+------+--------+---------+----------+

2 rows in set (0.00 sec)

mysql>



**Assignment 1 (Queries and Answers)**

**1) Create employee table with following schema: employee(emp\_id, emp\_name, salary, designation, company\_name).**

mysql> create table employee(emp\_id int, emp\_name varchar(30), salary int, designation varchar(30), company\_name varchar(30));

Query OK, 0 rows affected (0.23 sec)

**2)Add city column into employee.**

mysql> alter table employee add city varchar(30);

Query OK, 0 rows affected (0.08 sec)

Records: 0 Duplicates: 0 Warnings: 0

**3)Insert at least 10 records with meaningful data.**

mysql> insert into employee

values(1,"Mr.Sohail",10000,"Supervisor","KPIT","Kolhapur");

Query OK, 1 row affected (0.05 sec)

mysql> insert into employee values(2,"Mr.Vicky",20000,"Developer","TCS","Mumbai");

Query OK, 1 row affected (0.07 sec)

mysql> insert into employee values(3,"Mr.Ram",25000,"Ass. Developer","KPIT","Pune");

Query OK, 1 row affected (0.20 sec)

mysql> insert into employee values(4,"Mr.Raju",45000,"Ass. Developer","Infosys","Pune");

Query OK, 1 row affected (0.05 sec)

mysql> insert into employee values(5,"Mr.Akash",25000,"Asso. Software Engg","Capgemini","Noida");

Query OK, 1 row affected (0.07 sec)

mysql> insert into employee values(6,"Mr.Prathamesh",57000,"Software Engg","Capgemini","Noida");

Query OK, 1 row affected (0.12 sec)

mysql> insert into employee values(7,"Mrs.Swati",78000,"Project Engg","Adobe","Jaipur");

Query OK, 1 row affected (0.05 sec)

mysql> insert into employee values(8,"Mrs.Priya",96000,"Supervisor","Vodafone Solutions","Pune");

Query OK, 1 row affected (0.11 sec)

mysql> insert into employee values(9,"Mrs.Sonali",37800,"Technical Engg","Samsung","Kolhapur");

Query OK, 1 row affected (0.05 sec)

mysql> insert into employee values(10,"Mrs.Aarti",83000,"Programmer Analyst","TCS","Mumbai");

Query OK, 1 row affected (0.10 sec)

mysql> select \* from employee;

+--------+---------------+--------+---------------------+--------------------+----------+

| emp\_id | emp\_name | salary | designation | company\_name | city |

+--------+---------------+--------+---------------------+--------------------+----------+

| 1 | Mr.Sohail | 10000 | Supervisor | KPIT | Kolhapur |

| 2 | Mr.Vicky | 20000 | Developer | TCS | Mumbai |

| 3 | Mr.Ram | 25000 | Ass. Developer | KPIT | Pune |

| 4 | Mr.Raju | 45000 | Ass. Developer | Infosys | Pune |

| 5 | Mr.Akash | 25000 | Asso. Software Engg | Capgemini | Noida |

| 6 | Mr.Prathamesh | 57000 | Software Engg | Capgemini | Noida |

| 7 | Mrs.Swati | 78000 | Project Engg | Adobe | Jaipur |

| 8 | Mrs.Priya | 96000 | Supervisor | Vodafone Solutions | Pune |

| 9 | Mrs.Sonali | 37800 | Technical Engg | Samsung | Kolhapur |

| 10 | Mrs.Aarti | 83000 | Programmer Analyst | TCS | Mumbai |

+--------+---------------+--------+---------------------+--------------------+----------+

10 rows in set (0.00 sec)

**4)Display all employees with their salary.**

mysql> select emp\_id,emp\_name,salary from employee;

+--------+---------------+--------+

| emp\_id | emp\_name | salary |

+--------+---------------+--------+

| 1 | Mr.Sohail | 10000 |

| 2 | Mr.Vicky | 20000 |

| 3 | Mr.Ram | 25000 |

| 4 | Mr.Raju | 45000 |

| 5 | Mr.Akash | 25000 |

| 6 | Mr.Prathamesh | 57000 |

| 7 | Mrs.Swati | 78000 |

| 8 | Mrs.Priya | 96000 |

| 9 | Mrs.Sonali | 37800 |

| 10 | Mrs.Aarti | 83000 |

+--------+---------------+--------+

10 rows in set (0.00 sec)

**5)Find the name of the employee along with their id.**

mysql> select emp\_id,emp\_name from employee;

+--------+---------------+

| emp\_id | emp\_name |

+--------+---------------+

| 1 | Mr.Sohail |

| 2 | Mr.Vicky |

| 3 | Mr.Ram |

| 4 | Mr.Raju |

| 5 | Mr.Akash |

| 6 | Mr.Prathamesh |

| 7 | Mrs.Swati |

| 8 | Mrs.Priya |

| 9 | Mrs.Sonali |

| 10 | Mrs.Aarti |

+--------+---------------+

10 rows in set (0.00 sec)

**6)Find employees who are living in “Kolhapur”.**

mysql> select emp\_id,emp\_name,city from employee where city="Kolhapur";

+--------+------------+----------+

| emp\_id | emp\_name | city |

+--------+------------+----------+

| 1 | Mr.Sohail | Kolhapur |

| 9 | Mrs.Sonali | Kolhapur |

+--------+------------+----------+

2 rows in set (0.00 sec))

**7)Find name of the employees whose salary is >10000.**

mysql> select emp\_name,salary from employee where salary>10000;

+---------------+--------+

| emp\_name | salary |

+---------------+--------+

| Mr.Vicky | 20000 |

| Mr.Ram | 25000 |

| Mr.Raju | 45000 |

| Mr.Akash | 25000 |

| Mr.Prathamesh | 57000 |

| Mrs.Swati | 78000 |

| Mrs.Priya | 96000 |

| Mrs.Sonali | 37800 |

| Mrs.Aarti | 83000 |

+---------------+--------+

9 rows in set (0.00 sec)

**8)Find name of the employees whose salary is <15000.**

mysql> select emp\_name,salary from employee where salary<15000;

+-----------+--------+

| emp\_name | salary |

+-----------+--------+

| Mr.Sohail | 10000 |

+-----------+--------+

1 row in set (0.00 sec)

**9)Update designation of employee 4 to "Technical Engineer".**

mysql> update employee set designation="Techical Engineer" where emp\_id=4;

Query OK, 0 rows affected (0.00 sec)

Rows matched: 1 Changed: 0 Warnings: 0

mysql> select \* from employee;

+--------+---------------+--------+---------------------+--------------------+----------+

| emp\_id | emp\_name | salary | designation | company\_name | city |

+--------+---------------+--------+---------------------+--------------------+----------+

| 1 | Mr.Sohail | 10000 | Supervisor | KPIT | Kolhapur |

| 2 | Mr.Vicky | 20000 | Developer | TCS | Mumbai |

| 3 | Mr.Ram | 25000 | Ass. Developer | KPIT | Pune |

| 4 | Mr.Raju | 45000 | Techical Engineer | Infosys | Pune |

| 5 | Mr.Akash | 25000 | Asso. Software Engg | Capgemini | Noida |

| 6 | Mr.Prathamesh | 57000 | Software Engg | Capgemini | Noida |

| 7 | Mrs.Swati | 78000 | Project Engg | Adobe | Jaipur |

| 8 | Mrs.Priya | 96000 | Supervisor | Vodafone Solutions | Pune |

| 9 | Mrs.Sonali | 37800 | Technical Engg | Samsung | Kolhapur |

| 10 | Mrs.Aarti | 83000 | Programmer Analyst | TCS | Mumbai |

+--------+---------------+--------+---------------------+--------------------+----------+

10 rows in set (0.00 sec)

**10)Delete the employees having designation "Supervisor".**

mysql> delete from employee where designation="Supervisor";

Query OK, 2 rows affected (0.06 sec)

mysql> select \* from employee;

+--------+---------------+--------+---------------------+--------------+----------+

| emp\_id | emp\_name | salary | designation | company\_name | city |

+--------+---------------+--------+---------------------+--------------+----------+

| 2 | Mr.Vicky | 20000 | Developer | TCS | Mumbai |

| 3 | Mr.Ram | 25000 | Ass. Developer | KPIT | Pune |

| 4 | Mr.Raju | 45000 | Techical Engineer | Infosys | Pune |

| 5 | Mr.Akash | 25000 | Asso. Software Engg | Capgemini | Noida |

| 6 | Mr.Prathamesh | 57000 | Software Engg | Capgemini | Noida |

| 7 | Mrs.Swati | 78000 | Project Engg | Adobe | Jaipur |

| 9 | Mrs.Sonali | 37800 | Technical Engg | Samsung | Kolhapur |

| 10 | Mrs.Aarti | 83000 | Programmer Analyst | TCS | Mumbai |

+--------+---------------+--------+---------------------+--------------+----------+

8 rows in set (0.00 sec)

**11)Increment the salary of employees by 5%.**

mysql> update employee set salary=salary+(salary\*5/100);

Query OK, 8 rows affected (0.06 sec)

Rows matched: 8 Changed: 8 Warnings: 0

mysql> select \* from employee;

+--------+---------------+--------+---------------------+--------------+----------+

| emp\_id | emp\_name | salary | designation | company\_name | city |

+--------+---------------+--------+---------------------+--------------+----------+

| 2 | Mr.Vicky | 21000 | Developer | TCS | Mumbai |

| 3 | Mr.Ram | 26250 | Ass. Developer | KPIT | Pune |

| 4 | Mr.Raju | 47250 | Techical Engineer | Infosys | Pune |

| 5 | Mr.Akash | 26250 | Asso. Software Engg | Capgemini | Noida |

| 6 | Mr.Prathamesh | 59850 | Software Engg | Capgemini | Noida |

| 7 | Mrs.Swati | 81900 | Project Engg | Adobe | Jaipur |

| 9 | Mrs.Sonali | 39690 | Technical Engg | Samsung | Kolhapur |

| 10 | Mrs.Aarti | 87150 | Programmer Analyst | TCS | Mumbai |

+--------+---------------+--------+---------------------+--------------+----------+

**12)Find the name of employee having salary between 18000 and 22000.**

mysql> select emp\_name,salary from employee where salary between 18000 and 22000;

+----------+--------+

| emp\_name | salary |

+----------+--------+

| Mr.Vicky | 21000 |

+----------+--------+

1 row in set (0.00 sec)

**13)Find the number of employees working in each company.**

mysql> select company\_name,count(\*) as number\_of\_employees from employee group by company\_name;

+--------------+---------------------+

| company\_name | number\_of\_employees |

+--------------+---------------------+

| TCS | 2 |

| KPIT | 1 |

| Infosys | 1 |

| Capgemini | 2 |

| Adobe | 1 |

| Samsung | 1 |

+--------------+---------------------+

6 rows in set (0.00 sec)

**14)Find employees having same salary as "Ram".**

mysql> select emp\_name,salary from employee where salary=(select salary from employee where emp\_name="Mr.Ram");

+----------+--------+

| emp\_name | salary |

+----------+--------+

| Mr.Ram | 26250 |

| Mr.Akash | 26250 |

+----------+--------+

2 rows in set (0.00 sec)

**15)Find the employee having maximum salary.**

mysql> select emp\_name,salary from employee where salary=(select max(salary) from employee);

+-----------+--------+

| emp\_name | salary |

+-----------+--------+

| Mrs.Aarti | 87150 |

+-----------+--------+

1 row in set (0.00 sec)

**16)Find company having highest average salary.**

mysql> select company\_name,avg(salary) as max\_avg\_salary from employee group by company\_name order by avg(salary) desc limit 1;

+--------------+----------------+

| company\_name | max\_avg\_salary |

+--------------+----------------+

| Adobe | 81900.0000 |

+--------------+----------------+

1 row in set (0.00 sec)

**17)Find the employee having city name starting with letter "M".**

mysql> select emp\_name,city from employee where city like 'M%';

+-----------+--------+

| emp\_name | city |

+-----------+--------+

| Mr.Vicky | Mumbai |

| Mrs.Aarti | Mumbai |

+-----------+--------+

2 rows in set (0.00 sec))

**18)Update city of employee with id=2 to "Kolhapur".**

mysql> update employee set city="Kolhapur" where emp\_id=2;

Query OK, 1 row affected (0.15 sec)

Rows matched: 1 Changed: 1 Warnings: 0

mysql> select \* from employee;

+--------+---------------+--------+---------------------+--------------+----------+

| emp\_id | emp\_name | salary | designation | company\_name | city |

+--------+---------------+--------+---------------------+--------------+----------+

| 2 | Mr.Vicky | 21000 | Developer | TCS | Kolhapur |

| 3 | Mr.Ram | 26250 | Ass. Developer | KPIT | Pune |

| 4 | Mr.Raju | 47250 | Techical Engineer | Infosys | Pune |

| 5 | Mr.Akash | 26250 | Asso. Software Engg | Capgemini | Noida |

| 6 | Mr.Prathamesh | 59850 | Software Engg | Capgemini | Noida |

| 7 | Mrs.Swati | 81900 | Project Engg | Adobe | Jaipur |

| 9 | Mrs.Sonali | 39690 | Technical Engg | Samsung | Kolhapur |

| 10 | Mrs.Aarti | 87150 | Programmer Analyst | TCS | Mumbai |

+--------+---------------+--------+---------------------+--------------+----------+

8 rows in set (0.00 sec)

**19)Find name of the employees who are living in the same city as "Mr. Raju".**

mysql> select emp\_name,city from employee where city=(select city from employee where emp\_name="Mr.Raju");

+----------+------+

| emp\_name | city |

+----------+------+

| Mr.Ram | Pune |

| Mr.Raju | Pune |

+----------+------+

2 rows in set (0.00 sec)

**Assignment 2: Part 1.1 – Database Implementation for Horizontal Fragmentation**

R67@Sohail-OptiPlex-5090:~$ mysql -u root -p

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 9

Server version: 8.0.30-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database employee;

Query OK, 1 row affected (0.00 sec)

mysql> use employee;

Database changed

mysql> create table emp\_server(emp\_id int primary key,emp\_name varchar(30), emp\_salary float, emp\_city varchar(30));

Query OK, 0 rows affected (0.02 sec)

mysql> insert into emp\_server values(100,"Aditi",65000,"Mumbai");

Query OK, 1 row affected (0.02 sec)

mysql> insert into emp\_server values(101,"Ram",60000,"Kolhapur");

Query OK, 1 row affected (0.01 sec)

mysql> insert into emp\_server values(102,"Rahul",35000,"Mumbai");

Query OK, 1 row affected (0.01 sec)

mysql> insert into emp\_server values(103,"Vicky",40000,"Pune");

Query OK, 1 row affected (0.01 sec)

mysql> insert into emp\_server values(104,"Anuj",10000,"Mumbai");

Query OK, 1 row affected (0.01 sec)

mysql> insert into emp\_server values(105,"Faizan",27500,"Nagpur");

Query OK, 1 row affected (0.01 sec)

mysql> select \* from emp\_server;

+--------+----------+------------+----------+

| emp\_id | emp\_name | emp\_salary | emp\_city |

+--------+----------+------------+----------+

| 100 | Aditi | 65000 | Mumbai |

| 101 | Ram | 60000 | Kolhapur |

| 102 | Rahul | 35000 | Mumbai |

| 103 | Vicky | 40000 | Pune |

| 104 | Anuj | 10000 | Mumbai |

| 105 | Faizan | 27500 | Nagpur |

+--------+----------+------------+----------+

6 rows in set (0.00 sec)

mysql> create table emp\_client(emp\_id int primary key,emp\_name varchar(30), emp\_salary float, emp\_city varchar(30));

Query OK, 0 rows affected (0.02 sec)

mysql> select \* from emp\_client;

Empty set (0.01 sec)

/\*After executing server-side and client-side program of horizontal fragmentation\*/

mysql> select \* from emp\_client;

+--------+----------+------------+----------+

| emp\_id | emp\_name | emp\_salary | emp\_city |

+--------+----------+------------+----------+

| 100 | Sohail | 65000 | Mumbai |

| 102 | Rahul | 35000 | Mumbai |

| 104 | Anuj | 10000 | Mumbai |

+--------+----------+------------+----------+

3 rows in set (0.00 sec)

**Assignment 2: Part 2.1 – Database Implementation for Vertical Fragmentation**

R67@Sohail-OptiPlex-5090:~$ mysql -u root -p

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 9

Server version: 8.0.30-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database employee;

Query OK, 1 row affected (0.00 sec)

mysql> use employee;

Database changed

mysql> create table emp\_server(emp\_id int primary key,emp\_name varchar(30), emp\_salary float, emp\_city varchar(30));

Query OK, 0 rows affected (0.02 sec)

mysql> insert into emp\_server values(100,"Aditi",65000,"Mumbai");

Query OK, 1 row affected (0.02 sec)

mysql> insert into emp\_server values(101,"Ram",60000,"Kolhapur");

Query OK, 1 row affected (0.01 sec)

mysql> insert into emp\_server values(102,"Rahul",35000,"Mumbai");

Query OK, 1 row affected (0.01 sec)

mysql> insert into emp\_server values(103,"Vicky",40000,"Pune");

Query OK, 1 row affected (0.01 sec)

mysql> insert into emp\_server values(104,"Anuj",10000,"Mumbai");

Query OK, 1 row affected (0.01 sec)

mysql> insert into emp\_server values(105,"Faizan",27500,"Nagpur");

Query OK, 1 row affected (0.01 sec)

mysql> select \* from emp\_server;

+--------+----------+------------+----------+

| emp\_id | emp\_name | emp\_salary | emp\_city |

+--------+----------+------------+----------+

| 100 | Aditi | 65000 | Mumbai |

| 101 | Ram | 60000 | Kolhapur |

| 102 | Rahul | 35000 | Mumbai |

| 103 | Vicky | 40000 | Pune |

| 104 | Anuj | 10000 | Mumbai |

| 105 | Faizan | 27500 | Nagpur |

+--------+----------+------------+----------+

6 rows in set (0.00 sec)

mysql> create table emp\_client(emp\_id int primary key,emp\_name varchar(30),emp\_city varchar(30));

Query OK, 0 rows affected (0.02 sec)

mysql> select \* from emp\_client;

Empty set (0.01 sec)

/\*After executing server-side and client-side program of vertical fragmentation\*/

mysql> select \* from emp\_client;

+--------+----------+----------+

| emp\_id | emp\_name | emp\_city |

+--------+----------+----------+

| 100 | Sohail | Mumbai |

| 101 | Ram | Kolhapur |

| 102 | Rahul | Mumbai |

| 103 | Vicky | Pune |

| 104 | Anuj | Mumbai |

| 105 | Faizan | Nagpur |

+--------+----------+----------+

6 rows in set (0.00 sec)

**Assignment 3: Part 1.1 – Database Implementation for Semijoin Operation**

R67@Aditi-OptiPlex-390:~$ mysql -u root -p

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 4

Server version: 5.7.39-0ubuntu0.18.04.2 (Ubuntu)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database DYP\_Bank;

Query OK, 1 row affected (0.15 sec)

mysql> use DYP\_Bank;

Database changed

mysql> create table depositor(acc\_no int primary key,cust\_name varchar(30));

Query OK, 0 rows affected (1.46 sec)

mysql> create table account(acc\_no int primary key,branch varchar(30),balance float);

Query OK, 0 rows affected (0.40 sec)

mysql> create table temp1(acc\_no int primary key);

Query OK, 0 rows affected (0.47 sec)

mysql> create table temp2(acc\_no int primary key,branch varchar(30),balance float);

Query OK, 0 rows affected (0.37 sec)

mysql> create table resultset(acc\_no int primary key,cust\_name varchar(30),branch varchar(30),balance float);

Query OK, 0 rows affected (0.43 sec)

mysql> insert into depositor values(120,"Aditi");

Query OK, 1 row affected (0.15 sec)

mysql> insert into depositor values(121,"Rahul");

Query OK, 1 row affected (0.06 sec)

mysql> insert into depositor values(122,"Vicky");

Query OK, 1 row affected (0.06 sec)

mysql> insert into depositor values(123,"Ram");

Query OK, 1 row affected (0.05 sec)

mysql> insert into depositor values(124,"Samir");

Query OK, 1 row affected (0.05 sec)

mysql> insert into account values(120,"Rajarampuri",10000);

Query OK, 1 row affected (0.05 sec)

mysql> insert into account values(122,"Shahupuri",20000.20);

Query OK, 1 row affected (0.06 sec)

mysql> insert into account values(123,"Tarabai Park",25000.30);

Query OK, 1 row affected (0.06 sec)

mysql> insert into account values(125,"Kadamwadi",55000);

Query OK, 1 row affected (0.05 sec)

mysql> insert into account values(126,"Bawada",125000);

Query OK, 1 row affected (0.04 sec)

mysql> select \* from depositor;

+--------+-----------+

| acc\_no | cust\_name |

+--------+-----------+

| 120 | Aditi |

| 121 | Rahul |

| 122 | Vicky |

| 123 | Ram |

| 124 | Samir |

+--------+-----------+

5 rows in set (0.02 sec)

mysql> select \* from account;

+--------+--------------+---------+

| acc\_no | branch | balance |

+--------+--------------+---------+

| 120 | Rajarampuri | 10000 |

| 122 | Shahupuri | 20000.2 |

| 123 | Tarabai Park | 25000.3 |

| 125 | Kadamwadi | 55000 |

| 126 | Bawada | 125000 |

+--------+--------------+---------+

5 rows in set (0.01 sec)

mysql> select \* from temp1;

Empty set (0.02 sec)

mysql> select \* from temp2;

Empty set (0.03 sec)

mysql> select \* from resultset;

Empty set (0.04 sec)

--After executing Semijoin operation program

mysql> select \* from temp1;

+--------+

| acc\_no |

+--------+

| 120 |

| 121 |

| 122 |

| 123 |

| 124 |

+--------+

5 rows in set (0.02 sec)

--After executing Semijoin operation program

mysql> select \* from temp2;

+--------+--------------+---------+

| acc\_no | branch | balance |

+--------+--------------+---------+

| 120 | Rajarampuri | 10000 |

| 122 | Shahupuri | 20000.2 |

| 123 | Tarabai Park | 25000.3 |

+--------+--------------+---------+

3 rows in set (0.00 sec)

--After executing Semijoin operation program

mysql> select \* from resultset;

+--------+-----------+--------------+---------+

| acc\_no | cust\_name | branch | balance |

+--------+-----------+--------------+---------+

| 120 | Sohail | Rajarampuri | 10000 |

| 122 | Vicky | Shahupuri | 20000.2 |

| 123 | Ram | Tarabai Park | 25000.3 |

+--------+-----------+--------------+---------+

3 rows in set (0.00 sec)

**Advanced Database Systems**

**Question Bank**

1. **What are parallel systems? Compare parallel database architectures in detail with diagrams.**
2. **Explain data partitioning techniques used in parallel databases. Also give comparison between data partitioning techniques**.
3. Explain two phase commit (2PC) protocol in brief. Also explain how 2PC protocol handles failure of a participating site and failure of a coordinator.
4. **What are parallel systems? Explain Speedup and Scaleup in parallel systems with the help of diagram.**
5. What are the different types of distributed database system? Explain Semijoin strategy used in distributed query processing.
6. **Explain how to store data in distributed database systems.**
7. **What is stored procedure in PL/SQL? Give its advantages. Explain in detail, syntax to create stored procedure in PL/SQL with example.**
8. Describe oracle sequence. Explain sequence in Oracle with syntax and example.
9. **What is cursor in PL/SQL? Describe cursor attributes in detail. Write a PLSQL cursor to update salary of employee if salary is greater than 10000 then increment it by 6% otherwise by 5%.**
10. Explain in detail, syntax to create function in PL/SQL. Write PL/SQL function to find factorial of given number.
11. **What is trigger? Explain in detail, syntax to create trigger in oracle with example.**

**SemiJoin**

mysql -u root -p

Enter password: root

mysql> create database semijoin;

Query OK, 1 row affected (0.01 sec)

mysql> use semijoin;

Database changed

* **Table Depositor:**

mysql> create table depositor(acc\_no int(20),cust\_name varchar(20));

Query OK, 0 rows affected (0.27 sec)

mysql> insert into depositor values(101,"Sakshi");

Query OK, 1 row affected (0.05 sec)

mysql> insert into depositor values(102,"Rutika");

Query OK, 1 row affected (0.05 sec)

mysql> insert into depositor values(103,"Labdhi");

Query OK, 1 row affected (0.03 sec)

mysql> insert into depositor values(104,"Harsh");

Query OK, 1 row affected (0.04 sec)

mysql> insert into depositor values(105,"Chirag");

Query OK, 1 row affected (0.05 sec)

mysql> insert into depositor values(109,"Shreyash");

Query OK, 1 row affected (0.04 sec)

mysql> insert into depositor values(110,"ABC");

Query OK, 1 row affected (0.05 sec)

mysql> select \* from depositor;

+--------+-----------+

| acc\_no | cust\_name |

+--------+-----------+

| 101 | Sakshi |

| 102 | Rutika |

| 103 | Labdhi |

| 104 | Harsh |

| 105 | Chirag |

| 109 | Shreyash |

| 110 | ABC |

+--------+-----------+

7 rows in set (0.01 sec)

* **Table Account :**

mysql> create table account(acc\_no int(20),branch\_name varchar(20),balance int(20));

Query OK, 0 rows affected (0.24 sec)

mysql> insert into account values(101,"Kagal",10000);

Query OK, 1 row affected (0.04 sec)

mysql> insert into account values(102,"Kolhapur",20000);

Query OK, 1 row affected (0.04 sec)

mysql> insert into account values(103,"Sangali",15000);

Query OK, 1 row affected (0.03 sec)

mysql> insert into account values(104,"Pune",15000);

Query OK, 1 row affected (0.04 sec)

mysql> insert into account values(105,"Kagal",10000);

Query OK, 1 row affected (0.05 sec)

mysql> insert into account values(106,"Mumbai",10000);

\Query OK, 1 row affected (0.05 sec)

mysql> insert into account values(107,"Mumbai",10000);

Query OK, 1 row affected (0.04 sec)

mysql> insert into account values(108,"Pune",10000);

Query OK, 1 row affected (0.04 sec)

mysql> select \* from account;

+--------+-------------+---------+

| acc\_no | branch\_name | balance |

+--------+-------------+---------+

| 101 | Kagal | 10000 |

| 102 | Kolhapur | 20000 |

| 103 | Sangali | 15000 |

| 104 | Pune | 15000 |

| 105 | Kagal | 10000 |

| 106 | Mumbai | 10000 |

| 107 | Mumbai | 10000 |

| 108 | Pune | 10000 |

+--------+-------------+---------+

8 rows in set (0.00 sec)

* **Table temp1:**

mysql> create table temp1(acc\_no int(20));

Query OK, 0 rows affected (0.36 sec)

* **Table temp2:**

mysql> create table temp2(acc\_no int(20),branch\_name varchar(20),balance int(20));

Query OK, 0 rows affected (0.35 sec)

* **Table result:**

mysql> create table result(acc\_no int(20),cust\_name varchar(20),branch\_name varc

har(20),balance int(20));

Query OK, 0 rows affected (0.25 sec)

Program :

import java.sql.\*;

class semijoin

{

public static void main(String args[])

{

int an;

String cname;

String bname;

int bal;

try{

Class.forName("com.mysql.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/semijoin?useSSL=false","root","root");

Statement stmt = con.createStatement();

Statement stmt1 = con.createStatement();

Statement stmt2 = con.createStatement();

Statement stmt3 = con.createStatement();

Statement stmt4 = con.createStatement();

Statement stmt5 = con.createStatement();

ResultSet rs = stmt.executeQuery("SELECT acc\_no from depositor");

while(rs.next()){

an=rs.getInt(1);

stmt1.executeUpdate("INSERT INTO temp1 VALUES("+an+")");

}

ResultSet rs1 = stmt2.executeQuery("select \* from account natural join temp1");

while(rs1.next()){

an=rs1.getInt(1);

bname=rs1.getString(2);

bal = rs1.getInt(3);

stmt3.executeUpdate("insert into temp2 values("+an+",'"+bname+"',"+bal+")");

}

ResultSet rs2 = stmt4.executeQuery("select \* from depositor natural join temp2");

while(rs2.next()){

an=rs2.getInt(1);

cname=rs2.getString(2);

bname=rs2.getString(3);

bal = rs2.getInt(4);

stmt5.executeUpdate("insert into result values("+an+",'"+cname+"','"+bname+"',"+bal+")");

}

System.out.println("Semijoin Result");

ResultSet rs3 = stmt5.executeQuery("select \* from result");

while(rs3.next()){

an=rs3.getInt(1);

cname=rs3.getString(2);

bname=rs3.getString(3);

bal = rs3.getInt(4);

System.out.println(an+"\t"+cname+"\t"+bname+"\t"+bal);

}

con.close();

}catch(Exception e){

System.out.println(e);

}

}

}

**Output:**

**Java Terminal :**

ubuntu@ubuntu-OptiPlex-390:~/Desktop/BTech\_A\_64$ export CLASSPATH=$CLASSPATH:/usr/share/java/mysql-connector-java.jar

ubuntu@ubuntu-OptiPlex-390:~/Desktop/BTech\_A\_64$ javac semijoin.java

ubuntu@ubuntu-OptiPlex-390:~/Desktop/BTech\_A\_64$ java semijoinSemijoin Result

102 Rutika Kolhapur 20000

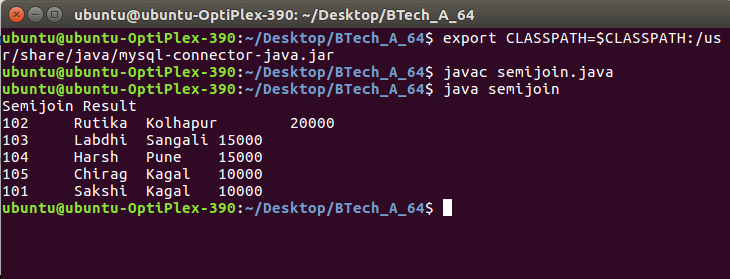
103 Labdhi Sangali 15000

104 Harsh Pune 15000

105 Chirag Kagal 10000

101 Sakshi Kagal 10000

ubuntu@ubuntu-OptiPlex-390:~/Desktop/BTech\_A\_64$



**MySQL Terminal :**

ubuntu@ubuntu-OptiPlex-390:~$ mysql -u root -p

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 15

Server version: 5.7.33-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select \* from temp1;

ERROR 1046 (3D000): No database selected

mysql> exit

Bye

ubuntu@ubuntu-OptiPlex-390:~$ clear

ubuntu@ubuntu-OptiPlex-390:~$ mysql -u root -p

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 16

Server version: 5.7.33-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| BTech\_64 |

| B\_3 |

| bank |

| btech38 |

| btecha38 |

| employee |

| mysql |

| performance\_schema |

| saksh |

| semijoin |

| student |

| sys |

+--------------------+

13 rows in set (0.00 sec)

mysql> use semijoin;

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Database changed

mysql> select \* from temp1;

+--------+

| acc\_no |

+--------+

| 102 |

| 103 |

| 104 |

| 105 |

| 109 |

| 110 |

| 101 |

+--------+

7 rows in set (0.00 sec)

mysql> select \* from temp2;

+--------+-------------+---------+

| acc\_no | branch\_name | balance |

+--------+-------------+---------+

| 101 | Kagal | 10000 |

| 102 | Kolhapur | 20000 |

| 103 | Sangali | 15000 |

| 104 | Pune | 15000 |

| 105 | Kagal | 10000 |

+--------+-------------+---------+

5 rows in set (0.00 sec)

mysql> select \* from result;

+--------+-----------+-------------+---------+

| acc\_no | cust\_name | branch\_name | balance |

+--------+-----------+-------------+---------+

| 102 | Rutika | Kolhapur | 20000 |

| 103 | Labdhi | Sangali | 15000 |

| 104 | Harsh | Pune | 15000 |

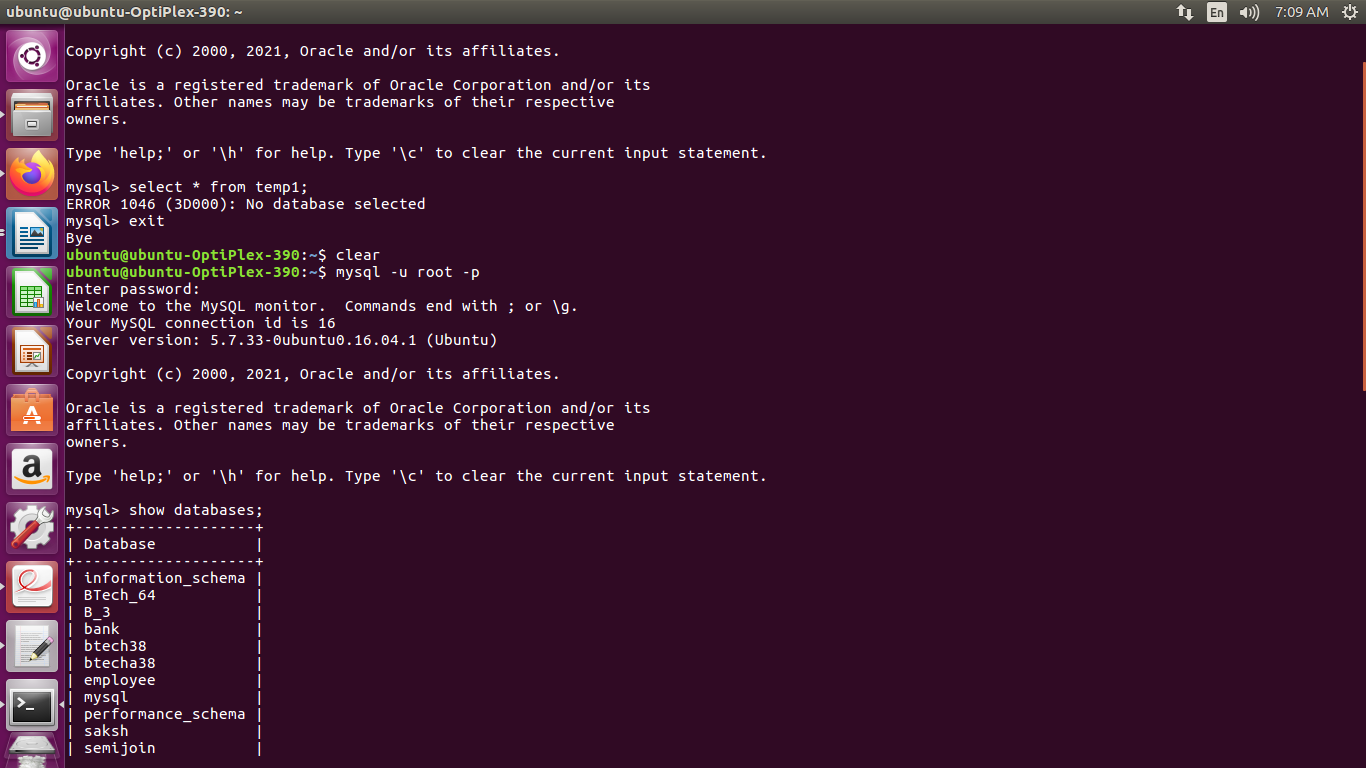
| 105 | Chirag | Kagal | 10000 |

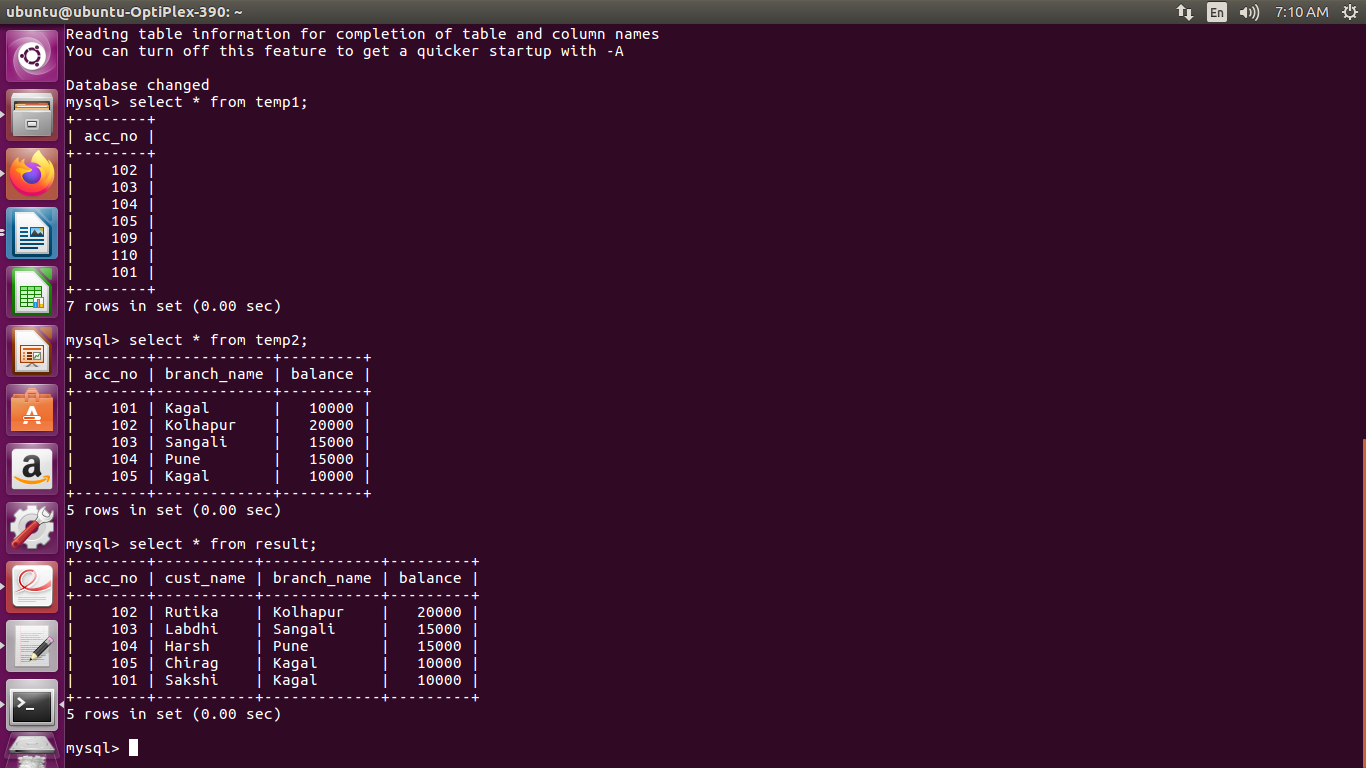
| 101 | Sakshi | Kagal | 10000 |

+--------+-----------+-------------+---------+

5 rows in set (0.00 sec)

mysql>





**Installation and Configuration case study of MySQL-JDBC**

**---------------------------------------------------------------------------------------------------------------------**

**Setting up MySQL or JDBC driver on Ubuntu:**

* sudo apt-get install libmysql-java
* export CLASSPATH=$CLASSPATH:/usr/share/java/mysql-connector-java.jar

**Simple Example JDBC-MySQL Connection**

import java.sql.\*;

class testdb{

public static void main(String args[]){

int rn=0;

String name;

try{

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection( "jdbc:mysql://localhost/stud","root","dypcet");

//here stud is database name, root is username and dypcet is password

Statement stmt=con.createStatement();

ResultSet rs=stmt.executeQuery("select \* from student");

while(rs.next())

{

rn= rs.getInt(1);

name= rs.getString(2);

System.out.println(“Rollno:”+rn+ “\t”+ “Name:”+nm);

}

con.close();

}catch(Exception e){ System.out.println(e);}

}

}

**Steps:**

1. Open Terminal

2. Open mysql prompt using

mysql –u root –p

Give password

3. create database stud;

4. use database stud;

5. create table student(rollno int, name varchar(20));

6. insert into student values(1, “Ram”);

Insert into student values (2, “Krishna”);

7. Open another Terminal

8. Install MySQL Library using following command

sudo apt-get install libmysql-java

9. Set Path to MySQL Library

export CLASSPATH=$CLASSPATH:/usr/share/java/mysql-connector-java.jar

10. Compile java program

javac testdb.java

11. Execute java program

java testdb

**Parallel Database Architectures**

**01.Shared Memory**

**-**All the processors share a common memory.

**-**In a shared-memory architecture, the processors and disks have access to a common memory, typically via a bus or through an interconnection network

-The benefit of shared memory is extremely efficient communication between processors—data in shared memory

- A processor can send messages to other processors much faster by using memory writes (which usually take less than a microsecond) than by sending a message through a communication mechanism

-Architecture is not scalable beyond 32 or 64 processors

-Adding more processors does not help after a point, since the processors will spend most of their time waiting for their turn on the bus to access memory.

-Shared-memory architectures usually have large memory caches at each processor

**2.Shared Disk**

**-**All the processors share a common set of disks

**-**In the shared-disk model, all processors can access all disks directly via an interconnection network, but the processors have private memories.

-*There is an advantages of shared disk over the shared memory first* is each processor has its own memory, the memory bus is not a bottleneck.*Second advaantage Is* it offers a cheap way to provide a degree of fault tolerance

-Shared-disk systems can scale to a somewhat larger number of processors, but communication across processors is slower

**3. Shared Nothing**

**-**The processors share neither a common memory nor common disk

-In a shared-nothing system, each node of the machine consists of a processor, memory, and one or more disks

-The processors at one node may communicate with another processor at another node by a high-speed interconnection network

-shared-nothing architectures are more scalable and can easily support a large number of processors.

-Shared-nothing multiprocessors can be scaled up to thousands of processors without interference

-The main drawbacks of shared-nothing systems are the costs of communication

**4. Hierarchical**

**-**This model is a hybrid of the preceding three architectures

-The hierarchical architecture combines the characteristics of shared-memory, shared-disk, and shared-nothing architectures

-Top level is a shared-nothing architecture – nodes connected by an interconnection network, and do not share disks or memory with each other



**I/O Parallelism**

**-**In its simplest form,I/O parallelism refers to reducing the time required to retrieve relations from disk by partitioning the relations over multiple disks

-Horizontal partitioning is the most common form of data partitioning technique in a parallel database

Several partitioning Techniques are

1. **Round-robin Partitioning**
2. **Hash partitioning**
3. **Range partitioning**

**1.Round-robin Partitioning**

-This strategy scans the relation in any order and sends the ith tuple to disk number Di mod n

- Round-robin scheme ensures that each disk has approximately the same number of tuples as the others.

-The scheme is ideally suited for applications that wish to read the entire relation sequentially for each query. With this scheme, both point queries and range queries are complicated to process, since each of the n disks must be used for the search.

**2.Hash partitioning**

-This declustering strategy designates one or more attributes from the given relation’s schema as the partitioning attributes

- A hash function is chosen whose range is {0, 1,..., n − 1}

- If the hash function returns i, then the tuple is placed on disk Di

-This scheme is best suited for point queries based on the partitioning attribute. For example, if a relation is partitioned on the telephone number attribute, then we can answer the query “Find the record of the employee with telephone number = 555-3333” by applying the partitioning hash function to 555-3333 and then searching that disk. Directing a query to a single disk saves the start-up cost of initiating a query on multiple disks, and leaves the other disks free to process other queries

**3.Range partitioning**

- This strategy distributes tuples by assigning contiguous attribute-value ranges to each disk.

- It chooses a partitioning attribute, A, and a partitioning vector [v0, v1,...,vn−2], such that, if i < j, then vi < vj . The relation is partitioned as follows: Consider a tuple t such that t[A] = x. If x < v0, then t goes on disk D0. If x ≥ vn−2, then t goes on disk Dn−1. If vi ≤ x < vi+1, then t goes on disk Di+1. For example, range partitioning with three disks numbered 0, 1, and 2 may assign tuples with values less than 5 to disk 0, values between 5 and 40 to disk 1, and values greater than 40 to disk 2.

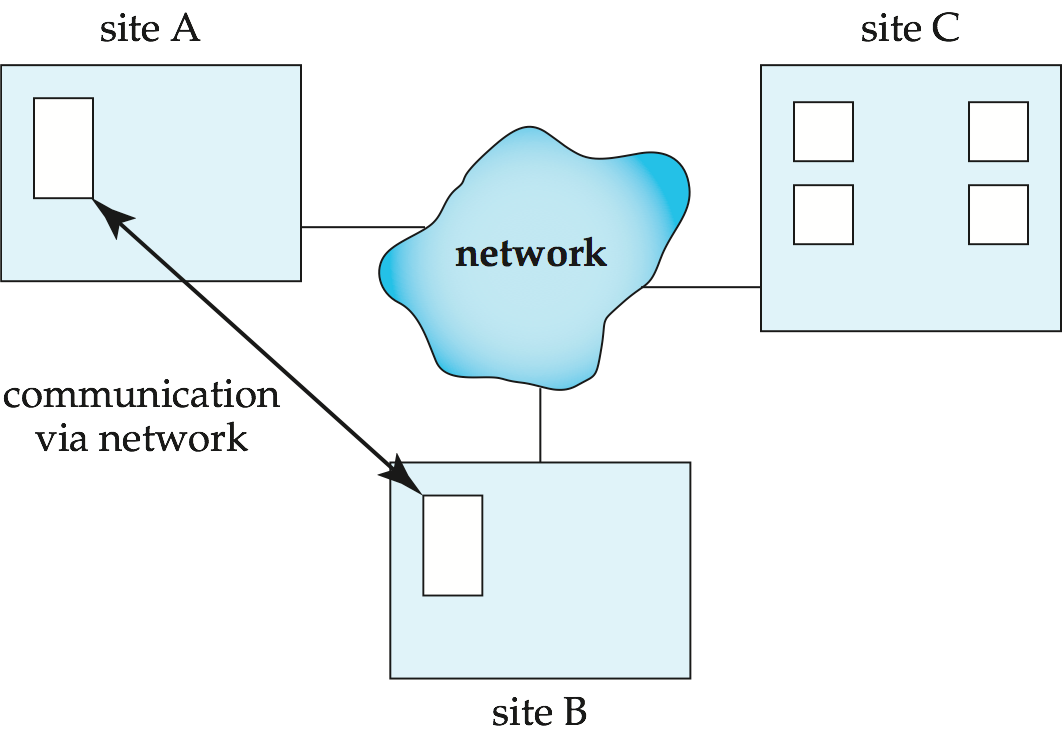
-This scheme is well suited for point and range queries on the partitioning attribute. For point queries, we can consult the partitioning vector to locate the disk where the tuple resides. For range queries, we consult the partitioning vector to find the range of disks on which the tuples may reside. In both cases, the search narrows to exactly those disks that might have any tuples of interest

**Distributed Systems**

-In a distributed database system, the database is stored on several computers.

-The computers in a distributed system communicate with one another through various communication media like high speed network or internet

-The computers in a distributed system are referred to by a number of different names, such as sites or nodes



Vertical fragmentation

* **Create Database :**

mysql> create database vertical;

Query OK, 1 row affected (0.00 sec)

mysql> show databases;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| B.Tech\_37 |

| BE\_A\_37 |

| BE\_Harsha |

| BTech\_78 |

| Bank |

| b1\_4 |

| btech39 |

| db\_name |

| mysql |

| performance\_schema |

| server\_DB |

| sm82 |

| sys |

| vertical |

+--------------------+

15 rows in set (0.00 sec)

mysql> use vertical;

Database changed

* **Create Table :**

mysql> create table employee(eid int(20),ename varchar(20),esalary int(20),ecity varchar(20));

Query OK, 0 rows affected (0.58 sec)

mysql> insert into employee values(101,"Sakshi",10000,"Kolhapur");

Query OK, 1 row affected (0.24 sec)

mysql> insert into employee values(102,"Rutika",20000,"Kolhapur");

Query OK, 1 row affected (0.36 sec)

mysql> insert into employee values(103,"Pranoti",30000,"Pune");

Query OK, 1 row affected (0.07 sec)

mysql> insert into employee values(104,"Aarya",40000,"Mumbai");

Query OK, 1 row affected (0.04 sec)

mysql> insert into employee values(105,"Teju",20000,"Mumbai");

Query OK, 1 row affected (0.39 sec)

mysql> create table emp1(eid int(20),ename varchar(20));

Query OK, 0 rows affected (0.25 sec)

**Server.java**

import java.util.\*;

import java.sql.\*;

import java.net.\*;

import java.io.\*;

public class server

{

public static void main(String args[])

{

DataInputStream dis;

DataOutputStream dos;

InputStream is;

OutputStream os;

Scanner s1=new Scanner(System.in);

try

{

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/vertical?useSSL=false","root","root");

Statement st=con.createStatement();

ServerSocket s=new ServerSocket(1520);

Socket s2=s.accept();

is=s2.getInputStream();

os=s2.getOutputStream();

dis=new DataInputStream(is);

dos=new DataOutputStream(os);

int count=0;

String rw=new String(dis.readUTF());

System.out.println(rw);

ResultSet rs1=st.executeQuery(rw);

while(rs1.next())

{

int id=rs1.getInt(1);

String nm=rs1.getString(2);

System.out.println(id+"\t"+nm);

count++;

}

dos.writeInt(count);

String nm; int id;

ResultSet rs2=st.executeQuery(rw);

while(rs2.next())

{

id=rs2.getInt(1);

nm=rs2.getString(2);

dos.writeInt(id);

dos.writeUTF(nm);

}

}

catch(Exception e)

{

System.out.println(e);

}

}

}

**Client.java**

import java.util.\*;

import java.sql.\*;

import java.net.\*;

import java.io.\*;

public class client

{

public static void main(String args[])

{

DataInputStream dis;

DataOutputStream dos;

InputStream is;

OutputStream os;

Statement st,st1,st2,st3;

Scanner s1=new Scanner(System.in);

try

{

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/vertical?useSSL=false","root","root");

st=con.createStatement();

Socket s=new Socket("127.0.0.1",1520);

is=s.getInputStream();

os=s.getOutputStream();

dis=new DataInputStream(is);

dos=new DataOutputStream(os);

System.out.println("Enter the query");

String rw=s1.nextLine();

dos.writeUTF(rw);

System.out.println("Vertical fragmentation");

int id,sal; String nm; String ect;

int count=dis.readInt();

for(int i=0;i<count;i++)

{

id= dis.readInt();

nm=dis.readUTF();

st.executeUpdate("insert into emp1 values("+id+",'"+nm+"')");

}

ResultSet rs1=st.executeQuery("select \* from emp1");

while(rs1.next())

{

int id1=rs1.getInt(1);

nm=rs1.getString(2);

System.out.println(id1+"\t"+nm);

}

}

catch(Exception e)

{ System.out.println(e);

}

}

}

**output:**

server.java

ubuntu@ubuntu-OptiPlex-3010:~/Desktop/BTech\_78/vertical$ export CLASSPATH=$CLASSPATH:/usr/share/java/mysql-connector-java.jar

ubuntu@ubuntu-OptiPlex-3010:~/Desktop/BTech\_78/vertical$ javac server.java

ubuntu@ubuntu-OptiPlex-3010:~/Desktop/BTech\_78/vertical$ java server

select eid,ename from employee;

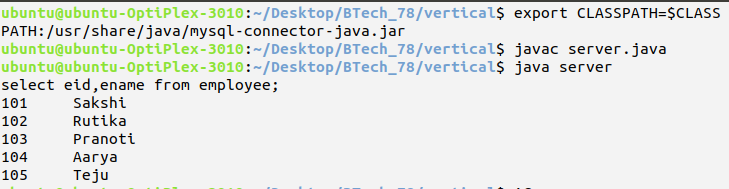
101 Sakshi

102 Rutika

103 Pranoti

104 Aarya

105 Teju



client.java

ubuntu@ubuntu-OptiPlex-3010:~/Desktop/BTech\_78/vertical$ export CLASSPATH=$CLASSPATH:/usr/share/java/mysql-connector-java.jar

ubuntu@ubuntu-OptiPlex-3010:~/Desktop/BTech\_78/vertical$ javac client.java

ubuntu@ubuntu-OptiPlex-3010:~/Desktop/BTech\_78/vertical$ java client

Enter the query

select eid,ename from employee;

Vertical fragmentation

101 Sakshi

102 Rutika

103 Pranoti

104 Aarya

105 Teju

ubuntu@ubuntu-OptiPlex-3010:~/Desktop/BTech\_78/vertical$



mysql :

mysql> select \* from employee;

+------+---------+---------+----------+

| eid | ename | esalary | ecity |

+------+---------+---------+----------+

| 101 | Sakshi | 10000 | Kolhapur |

| 102 | Rutika | 20000 | Kolhapur |

| 103 | Pranoti | 30000 | Pune |

| 104 | Aarya | 40000 | Mumbai |

| 105 | Teju | 20000 | Mumbai |

+------+---------+---------+----------+

5 rows in set (0.00 sec)

mysql> select \* from emp1;

+------+---------+

| eid | ename |

+------+---------+

| 101 | Sakshi |

| 102 | Rutika |

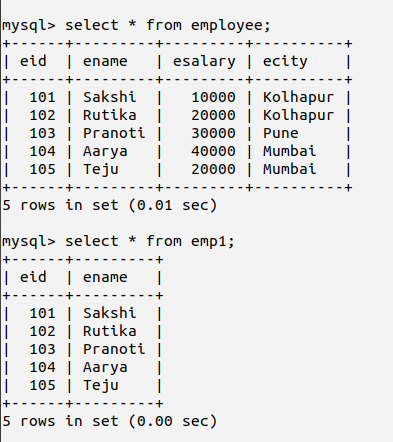
| 103 | Pranoti |

| 104 | Aarya |

| 105 | Teju |

+------+---------+

5 rows in set (0.00 sec)



**Assignment 2: Part 2.2 - Vertical Fragmentation Programs**

**Server-Side Program for Vertical Fragmentation:**

import java.util.\*;

import java.sql.\*;

import java.net.\*;

import java.io.\*;

public class server

{

public static void main(String args[])

{

DataInputStream dis;

DataOutputStream dos;

InputStream is;

OutputStream os;

Scanner s1=new Scanner(System.in);

try

{

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection("jdbc:mysql://localhost/employee","root","root");

Statement st=con.createStatement();

ServerSocket s=new ServerSocket(1520);

Socket s2=s.accept();

is=s2.getInputStream();

os=s2.getOutputStream();

dis=new DataInputStream(is);

dos=new DataOutputStream(os);

int id;

String name,city;

int count=0;

String rw=new String(dis.readUTF());

System.out.println(rw);

ResultSet rs1=st.executeQuery(rw);

while(rs1.next())

{

id=rs1.getInt(1);

name=rs1.getString(2);

city=rs1.getString(3);

System.out.println(id+"\t"+name+"\t"+city);

count++;

}

dos.writeInt(count);

ResultSet rs2=st.executeQuery(rw);

while(rs2.next())

{

id=rs2.getInt(1);

name=rs2.getString(2);

city=rs2.getString(3);

dos.writeInt(id);

dos.writeUTF(name);

dos.writeUTF(city);

}

}

catch(Exception e)

{

System.out.println(e);

}

}

}

**Output:**

select emp\_id,emp\_name,emp\_city from emp\_server;

100 Aditi Mumbai

101 Ram Kolhapur

102 Rahul Mumbai

103 Vicky Pune

104 Anuj Mumbai

105 Faizan Nagpur

**Client-Side Program for Vertical Fragmentation:**

import java.util.\*;

import java.sql.\*;

import java.net.\*;

import java.io.\*;

public class client

{

public static void main(String args[])

{

DataInputStream dis;

DataOutputStream dos;

InputStream is;

OutputStream os;

Statement st,st1,st2,st3;

Scanner s1=new Scanner(System.in);

try

{

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection("jdbc:mysql://localhost/employee","root","root");

st=con.createStatement();

Socket s=new Socket("127.0.0.1",1520);

is=s.getInputStream();

os=s.getOutputStream();

dis=new DataInputStream(is);

dos=new DataOutputStream(os);

System.out.println("Enter the query");

String rw=s1.nextLine();

dos.writeUTF(rw);

System.out.println("Vertical fragmentation");

int id;

String name,city;

int count=dis.readInt();

for(int i=0;i<count;i++)

{

id= dis.readInt();

name= dis.readUTF();

city= dis.readUTF();

st.executeUpdate("insert into emp\_client values("+id+",'"+name+"','"+city+"')");

}

ResultSet rs1=st.executeQuery("select \* from emp\_client");

while(rs1.next())

{

id=rs1.getInt(1);

name=rs1.getString(2);

city=rs1.getString(3);

System.out.println(id+"\t"+name+"\t"+city);

}

}

catch(Exception e)

{

System.out.println(e);

}

}

}

**Output:**

Enter the query

select emp\_id,emp\_name,emp\_city from emp\_server;

Vertical fragmentation

100 Aditi Mumbai

101 Ram Kolhapur

102 Rahul Mumbai

103 Vicky Pune

104 Anuj Mumbai

105 Faizan Nagpur