**Practical No.1**

**Title: Implementation of Data partitioning through Range and List partitioning**

**Range Partitioning Example:**

INPUT :-

create table emp (emp\_id int primary key, ename varchar(20), esalary int not null) partition by range (esalary)(partition P1 values less than(10000), partition P2 values less than(20000), partition P3 values less than(30000), partition P4 values less than(MAXVALUE)

);

select partition\_name, table\_name from user\_tab\_partitions;

select \* from emp partition(P1);



**To view data dictionary for Partitions:**

select partition\_name, table\_name from user\_tab\_partitions;

**Insert Values into table emp:**

insert into emp values(101,'Ajay',25000);

insert into emp values(102,'Raj',9000);

insert into emp values(103,'Anuj',20000);

insert into emp values(104,'Kamal',12500);

insert into emp values(105,'Sohan',15000);

insert into emp values(106,'Ankit',8000);

insert into emp values(107,'Prince',40000);

insert into emp values(108,'Shivani',22000);

insert into emp values(108,'Madhu',50000);

insert into emp values(108,'Amrita',70000);

Output:

**Display table emp data:**

select \* from emp;

**Display data from each partition individually:**

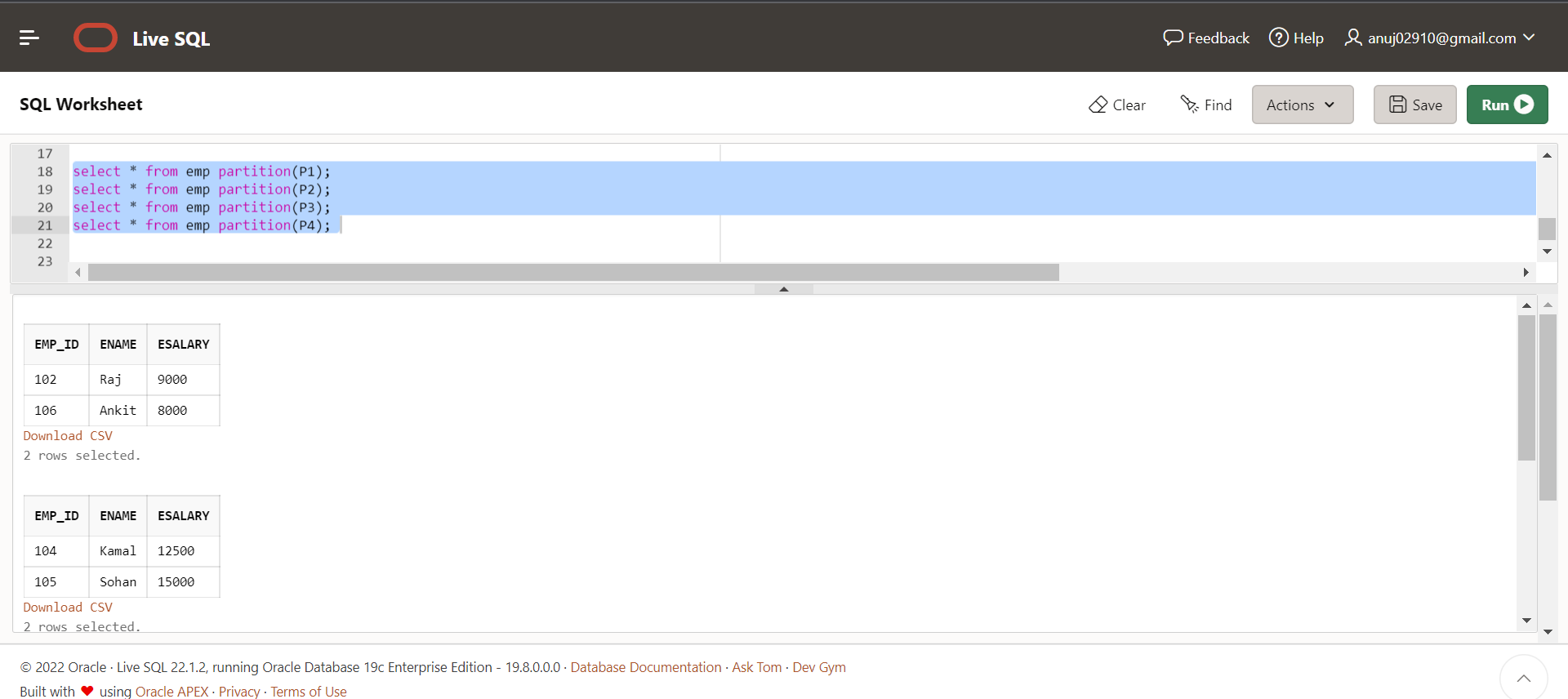
 Output:

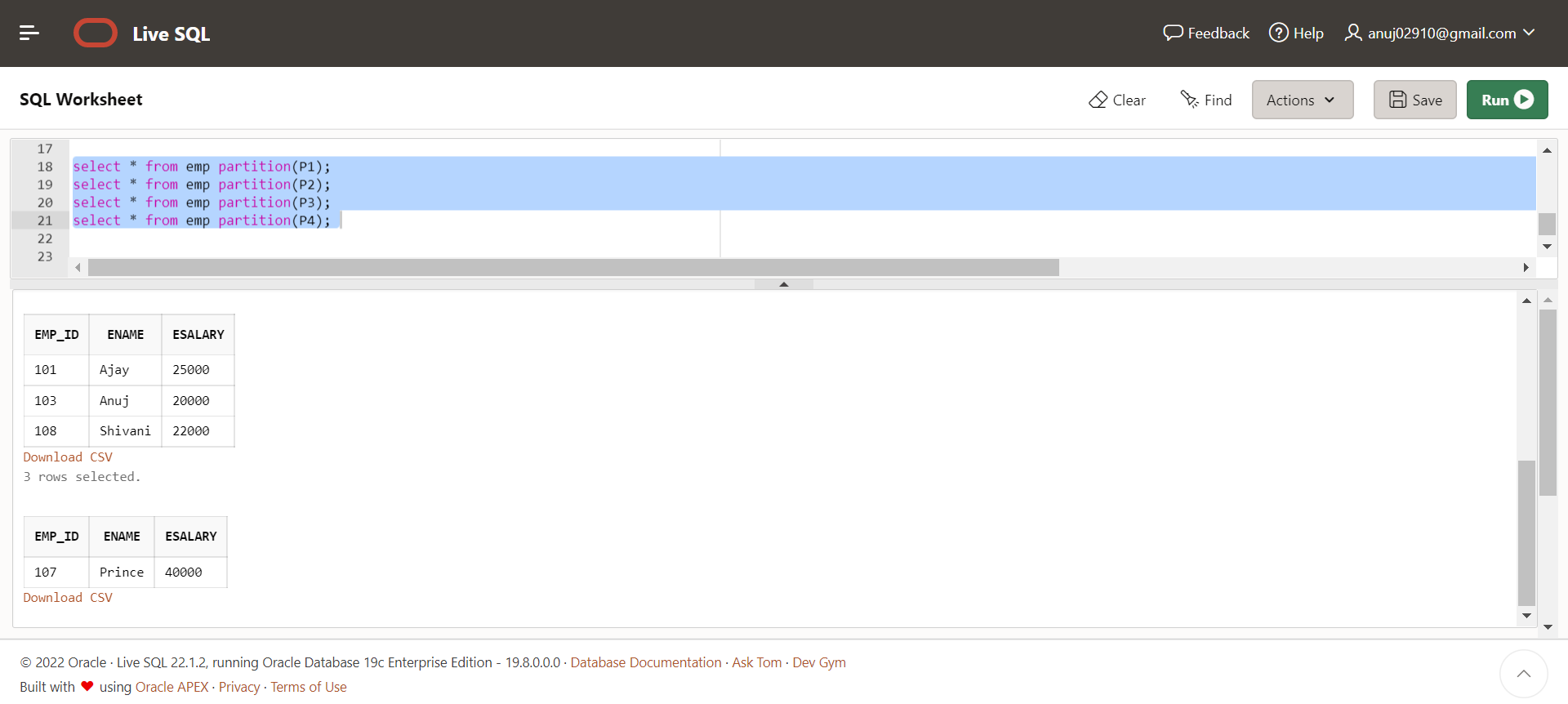
select \* from emp partition(P1);

select \* from emp partition(P2);

select \* from emp partition(P3);

select \* from emp partition(P4);





**List Partitioning**

|  |  |  |  |
| --- | --- | --- | --- |
| **List partitioning enables you to explicitly control how rows map to partitions. You do this by** | | |  |
| **specifying a list of discrete values for the partitioning key in the description for each partition.** | | | |
| **The advantage of list partitioning is that you can group and organize unordered and** | |  |  |
| **unrelated sets of data in a natural way.** |  |  |  |

CREATE TABLE Persons ( id INT PRIMARY KEY, name VARCHAR(10), city VarCHAR(10))

PARTITION BY LIST (city) (

PARTITION Karnataka VALUES ( 'Bengaluru','Kolar','Mysuru'),

PARTITION Maharashtra VALUES ( 'Mumbai','Pune','Nashik'),

PARTITION Madhya\_Pradesh VALUES ( 'Indore','Ujjain','Itarsi'),

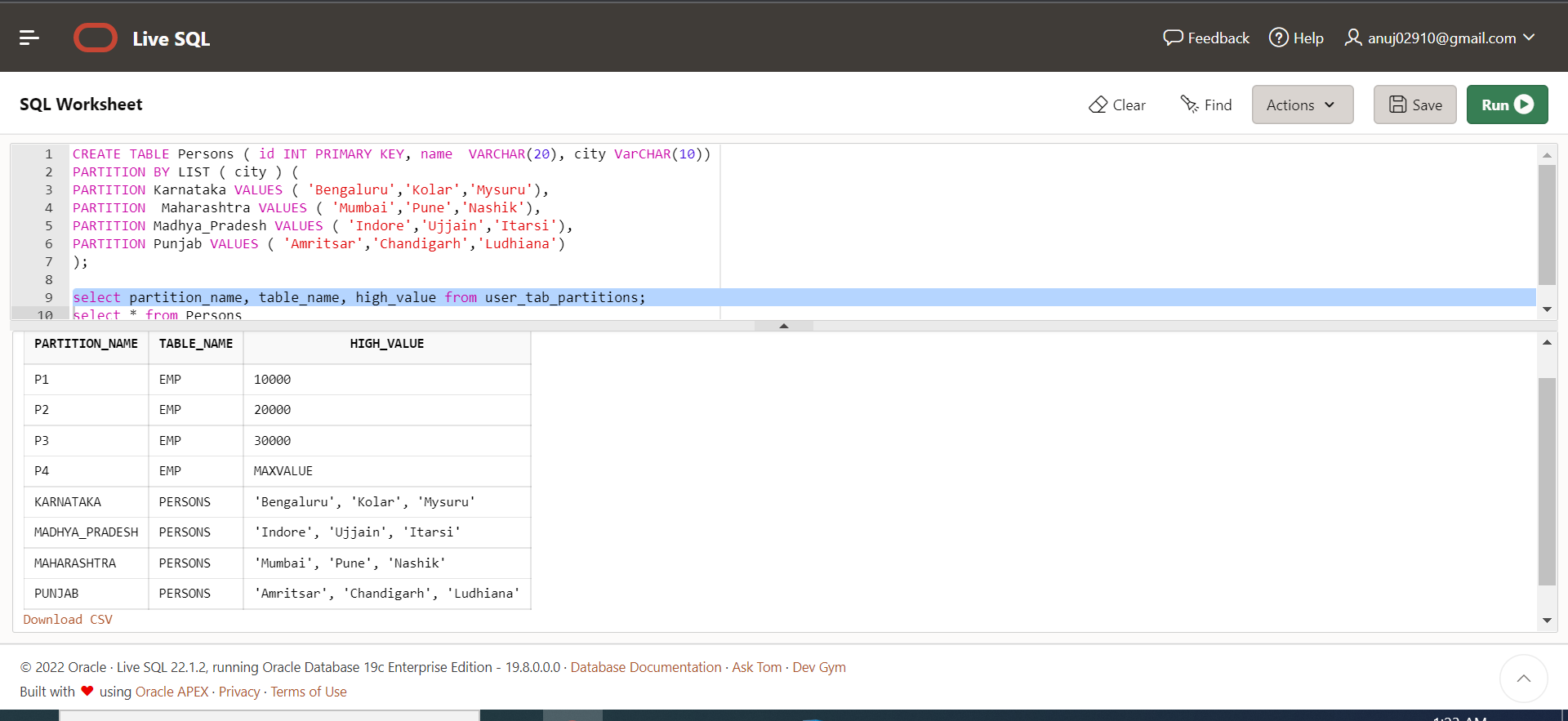
PARTITION Punjab VALUES ( 'Amritsar','Chandigarh','Ludhiana')

);

**To view data dictionary for Partitions:**

select partition\_name, table\_name, high\_value from user\_tab\_partitions;

Output:



**Insert values into table Persons:**

insert into Persons values(101,'Anuj','Kolar');

insert into Persons values(102,'Shivani','Mumbai');

insert into Persons values(103,'Riya','Pune');

insert into Persons values(104,'Jina','Bengaluru');

insert into Persons values(105,'Varsha','Amritsar');

insert into Persons values(106,'Shubham','Mysuru');

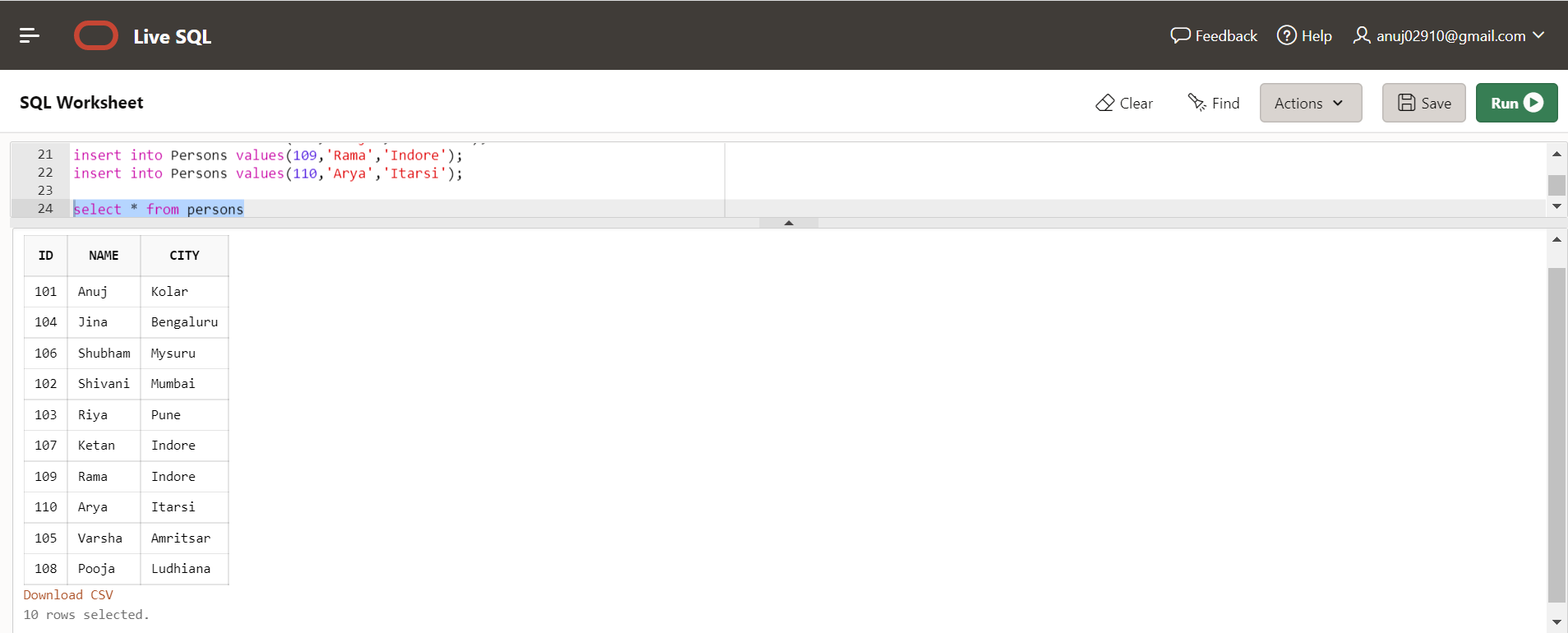
insert into Persons values(107,'Ketan','Indore');

insert into Persons values(108,'Pooja','Ludhiana');

insert into Persons values(109,'Rama','Indore');

insert into Persons values(110,'Arya','Itarsi');

Output:



**Display table Persons data:**

select \* from Persons;

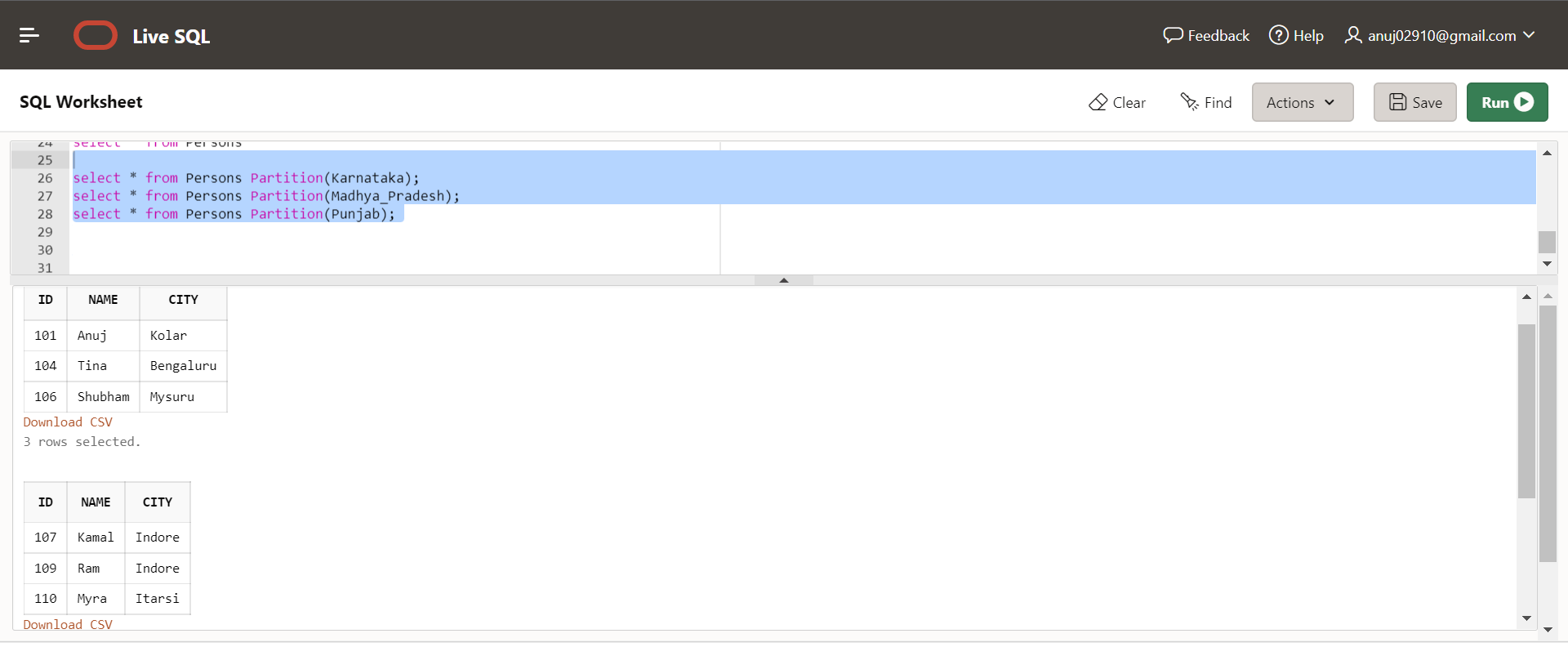
**Display data from each partition individually:**

select \* from Persons Partition(Karnataka);

select \* from Persons Partition(Madhya\_Pradesh);

select \* from Persons Partition(Punjab);

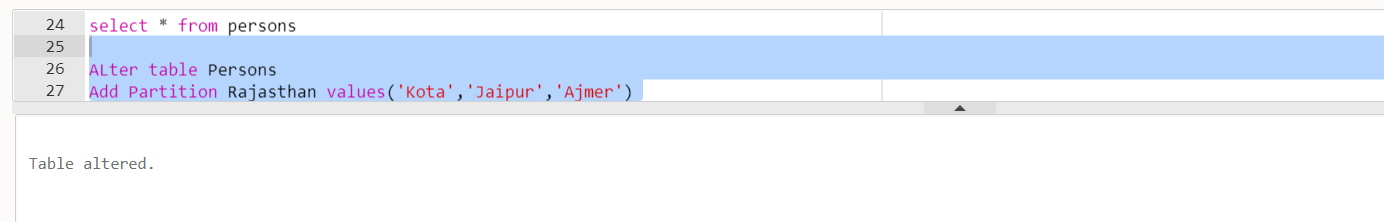
Output :-



**Add new Partition into existing Table:**

Alter table persons

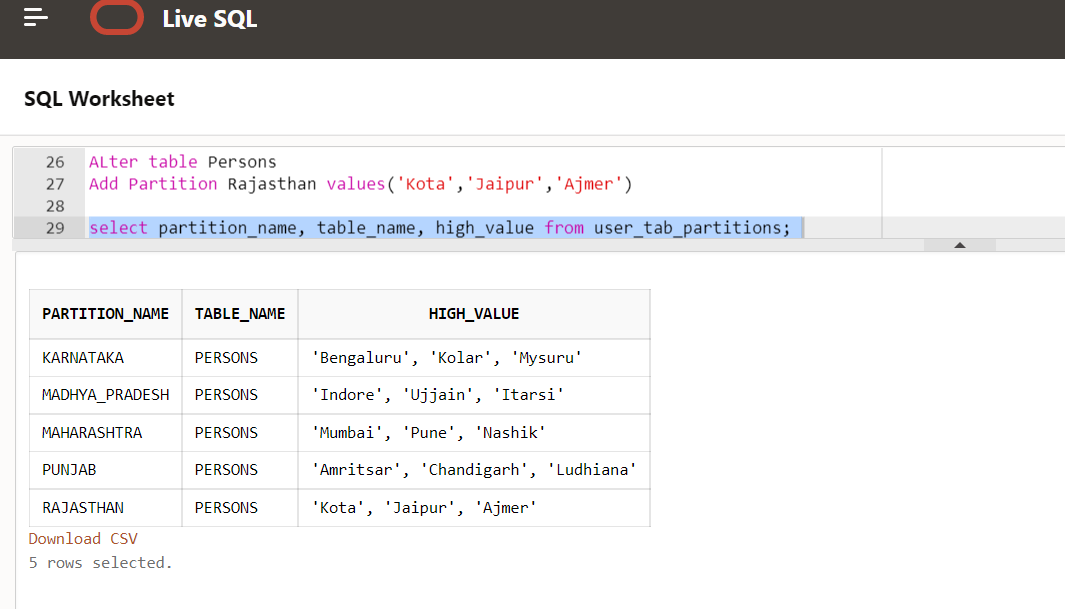
ADD Partition Rajasthan values('Ajmer','Jaipur','Kota','Jodhpur');



**To view data dictionary for Partitions:**

select partition\_name, table\_name, high\_value from user\_tab\_partitions;

Output:



**Insert data into table persons for new partition:**

insert into Persons values(111,'Rama','Kota');

insert into Persons values(112,'Manoj','Ajmer');

**Display data from new partition:-**

select \* from Persons Partition(Rajasthan);

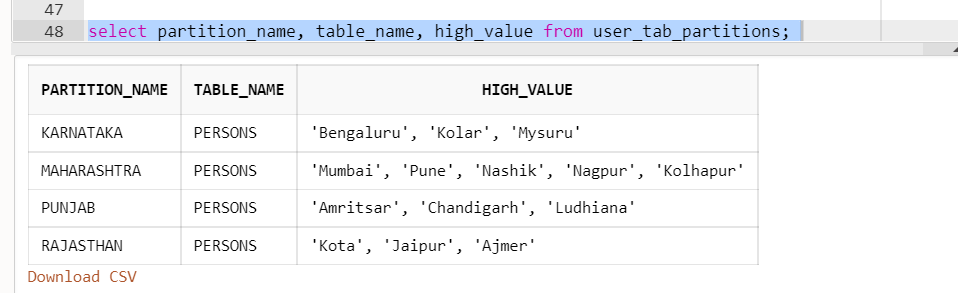
Output:



**Add Values in existing table:**

ALTER TABLE Persons

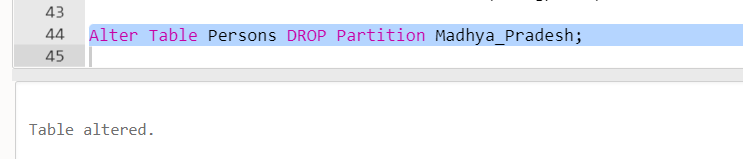
MODIFY PARTITION Maharashtra ADD VALUES ( ‘Nagpur’ );

Output:- 

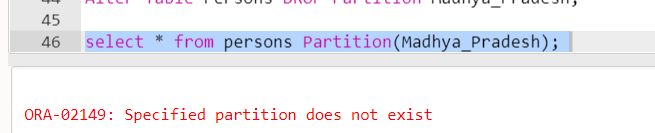
**Drop a Partition from a table Person:**

Alter Table Persons DROP Partition Madhya\_Pradesh;

Output:



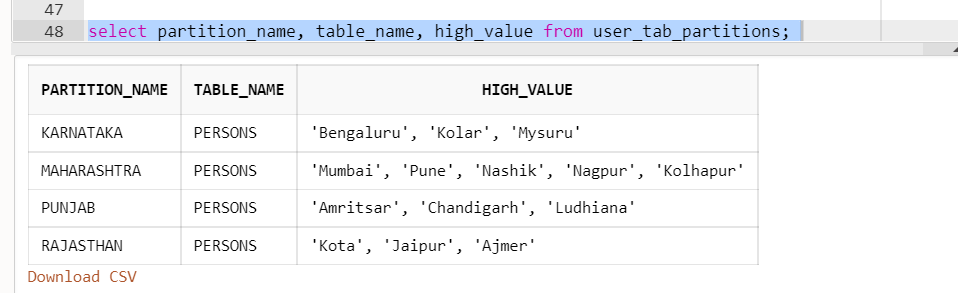
select \* from persons Partition(Madhya\_Pradesh);

Output: 

**To view current data dictionary for Partition:**

select partition\_name, table\_name, high\_value from user\_tab\_partitions;

Output:



**Example on List Partitioning with Default Partition:**

CREATE TABLE Employee ( id INT PRIMARY KEY, name VARCHAR(10), state\_code CHAR(2) )

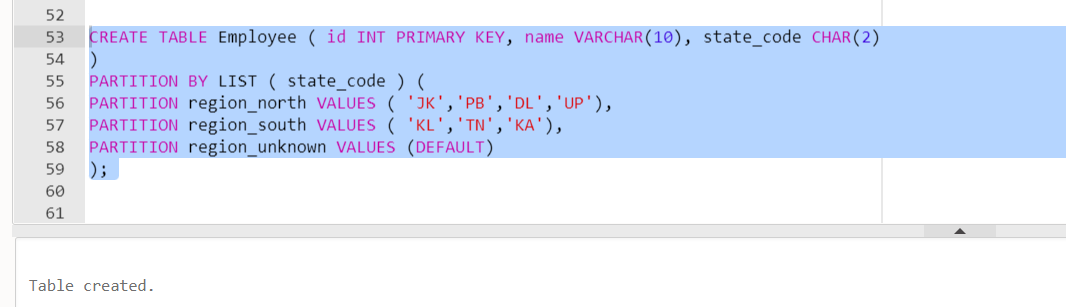
PARTITION BY LIST ( state\_code ) (

PARTITION region\_north VALUES ( 'JK','PB','DL','UP'),

PARTITION region\_south VALUES ( 'KL','TN','KA'),

PARTITION region\_unknown VALUES (DEFAULT)

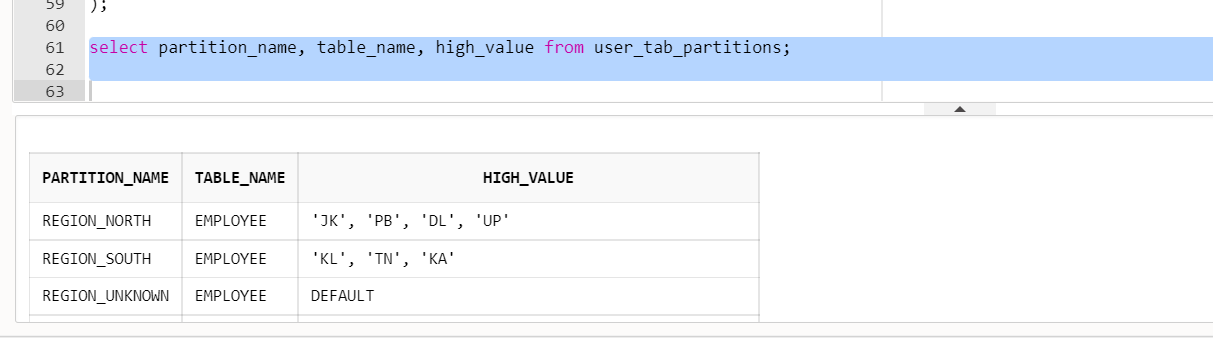
);



**To view data dictionary for Partitions:**

select partition\_name, table\_name, high\_val from user\_tab\_partitions;

Output:



**Insert data into Employee Table:**

INSERT INTO Employee VALUES (100,'Ameer','KA');

INSERT INTO Employee VALUES (101,'Suraj','PB');

INSERT INTO Employee VALUES (102,'Boby','JK');

INSERT INTO Employee VALUES (103,'Riyo','KL');

INSERT INTO Employee VALUES (104,'Arrva','DL');

INSERT INTO Employee VALUES (105, 'Pravesh', 'DL');

**Display data from created Partitions in the Table Employee:**

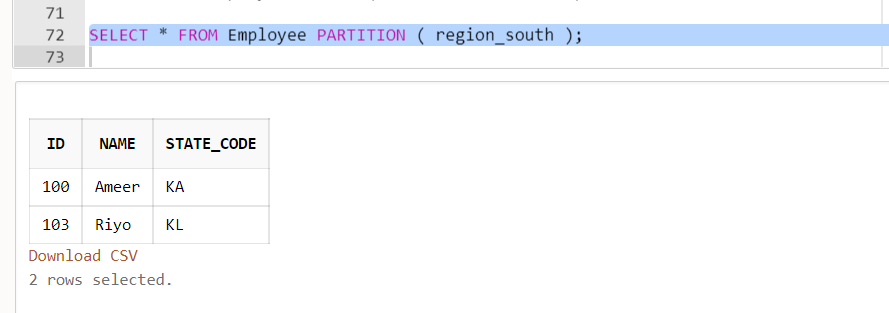
SELECT \* FROM Employee PARTITION ( region\_north );

Output:



SELECT \* FROM Employee PARTITION ( region\_south );

Output:



**Insert Some Records where state\_code values are not assigned to created partitions in table Employee:**

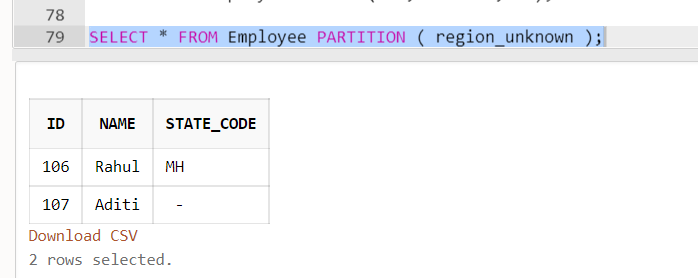
INSERT INTO Employee VALUES (106, 'Rahul', 'MH');

INSERT INTO Employee VALUES (107, 'Aditi', '');

**Here, these records are assigned to partition Region\_unknown:**

SELECT \* FROM Employee PARTITION ( region\_unknown );

Output:



**Practical-2**

.

**Aim: Implementation of Analytical queries like Roll\_UP, CUBE, FIRST, LAST, LEAD, LAG, RANK AND DENSE RANK**

Cube and Rollup

Simple Group Clause

create table emp\_data

(

empno number(10) not null,

ename varchar(10) not null,

job varchar(10) not null,

salary number(10),

comm number(10),

deptno number(10)

 )

desc emp\_data;

**To Insert the data into table:-**

insert into emp\_data values(1001,'Nikhil','Analyst',20000,1000,10);

insert into emp\_data values(1002,'Anish','Manager',30000,1000,20);

insert into emp\_data values(1003,'Om','Programmer',22000,1000,30);

insert into emp\_data values(1004,'Vrushabh','Clerk',35000,1000,40);

insert into emp\_data values(1005,'Kuldeep','Analyst',28000,null,50);

insert into emp\_data values(1006,'Jay','Manager',12000,null,40);

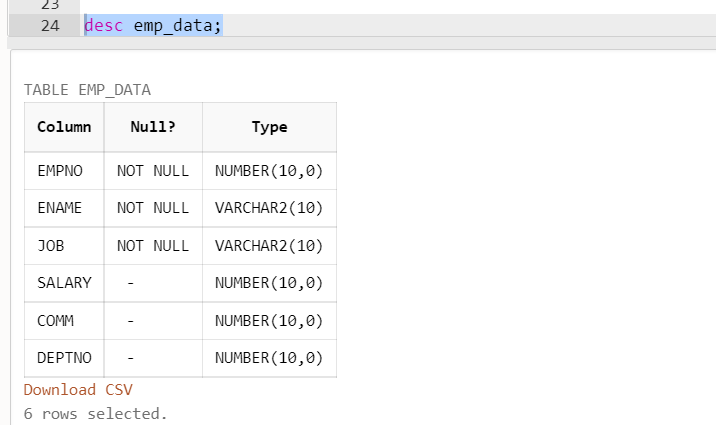
insert into emp\_data values(1007,'Pawan','Programmer',40000,null,30);

insert into emp\_data values(1008,'Sneha','Clerk',33000,null,20);

insert into emp\_data values(1009,'Ridhhi','Programmer',10000,1000,10);

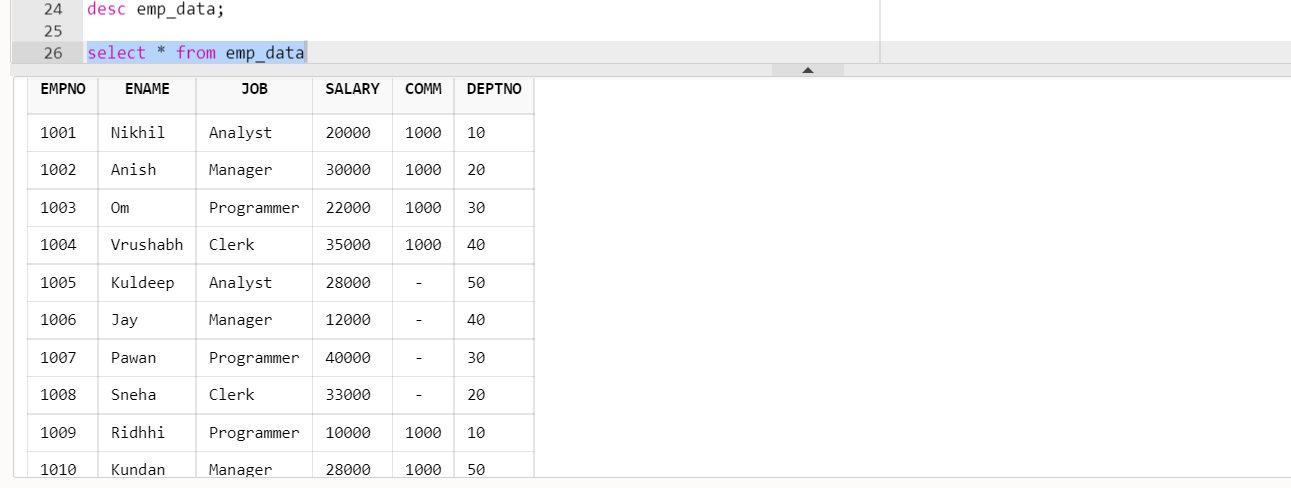
insert into emp\_data values(1010,'Kundan','Manager',28000,1000,50);

desc emp\_data;



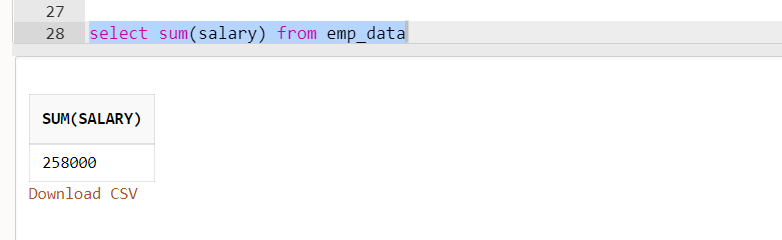
Output:

select \* from emp\_data



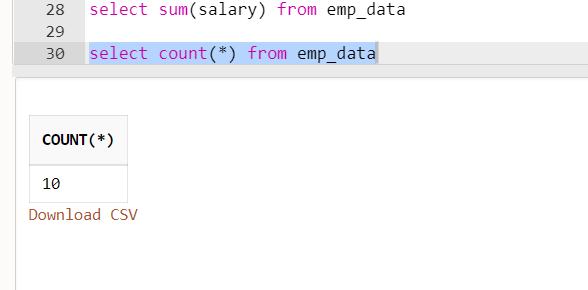
select sum(salary) from emp\_data

Output:



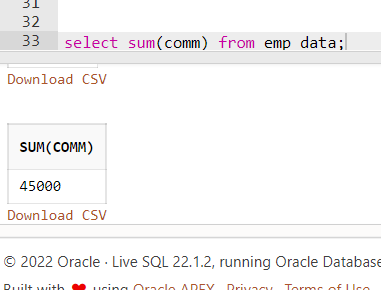
select count(\*) from emp\_data

Output:



select sum(comm) from emp\_data

Output:



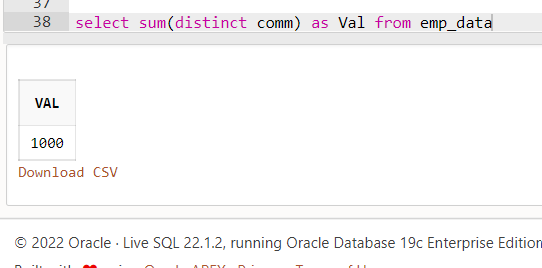
select sum(distinct comm) from emp\_data;

Output:



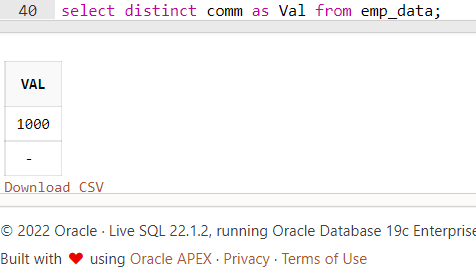
select sum(distinct comm) as Val from emp\_data

Output:



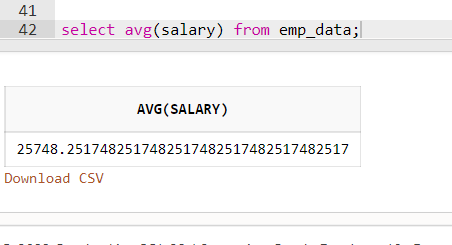
select distinct comm as Val from emp\_data

Output:



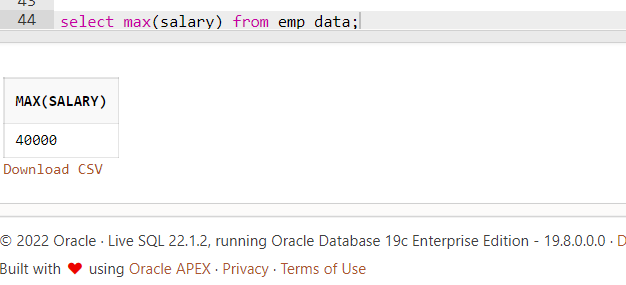
select avg(salary) from emp\_data;

Output:



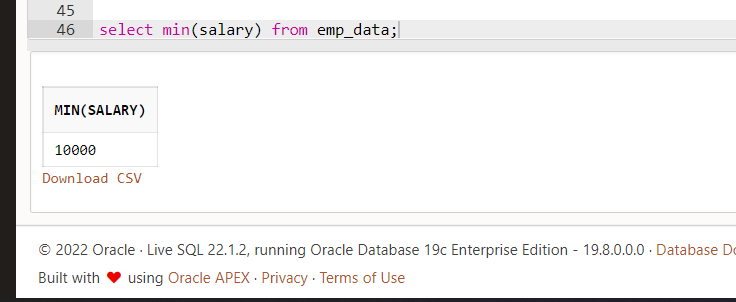
select max(salary) from emp\_data;

Output:



select min(salary) from emp\_data;

Output:



Order By

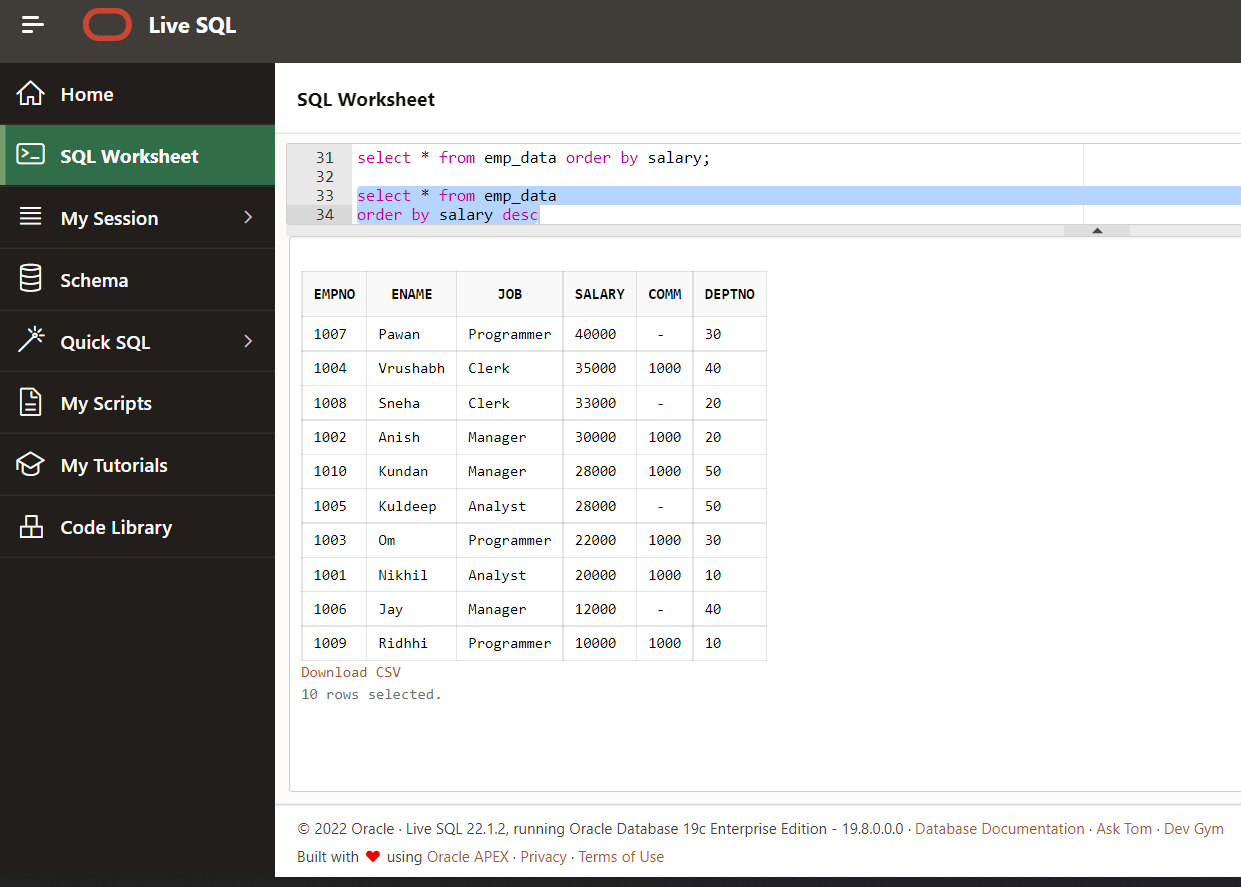
select \* from emp\_data order by salary;

Output:



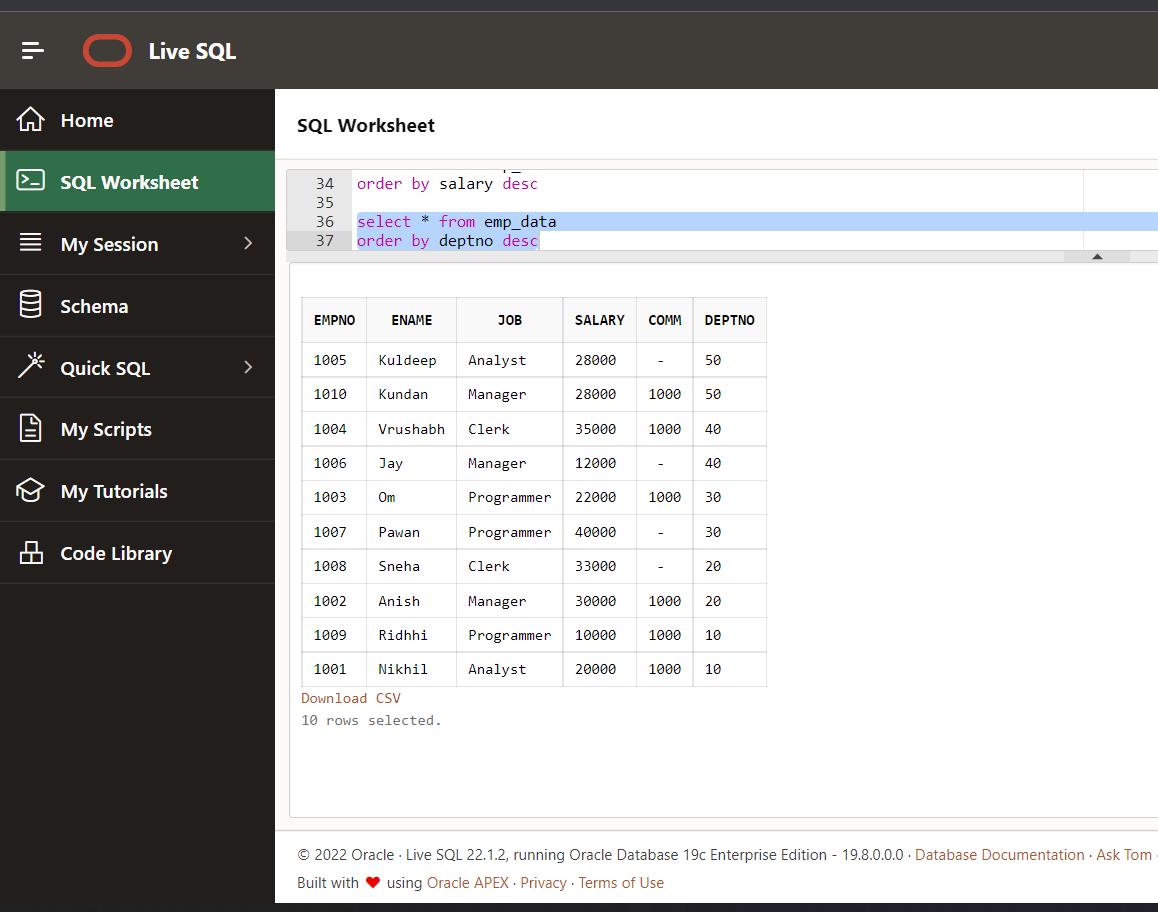
select \* from emp\_data order by salary desc;

Output:



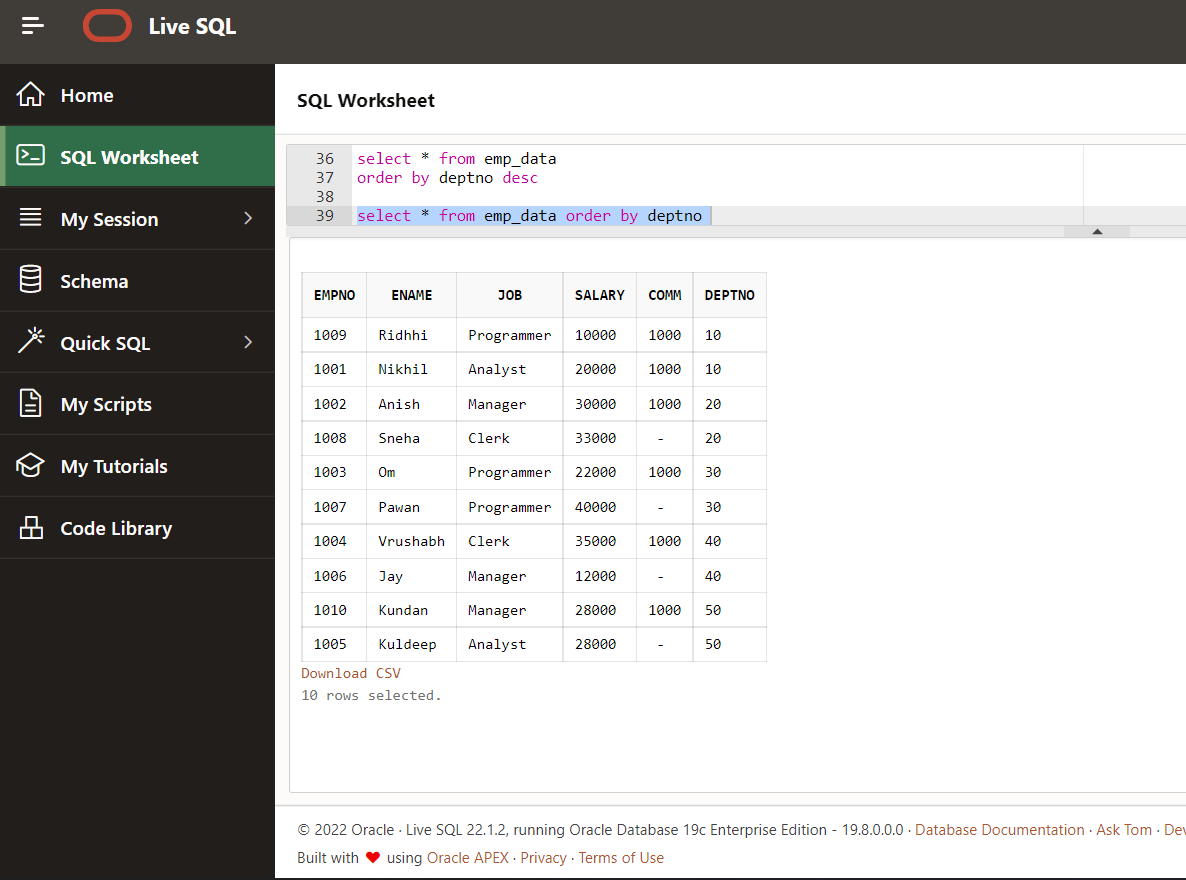
select \* from emp\_data order by deptno desc;

Output:



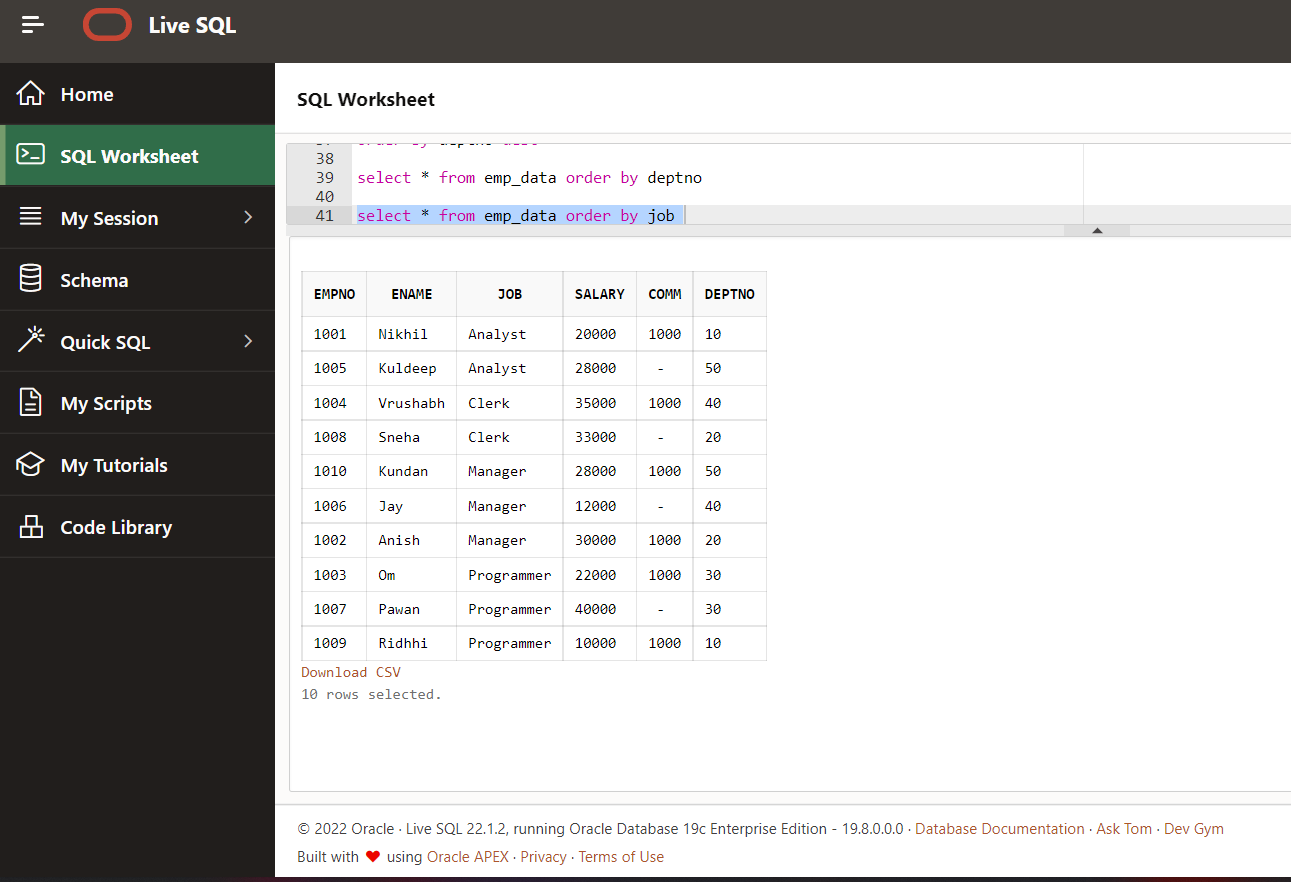
select \* from emp\_data order by deptno;

Output:



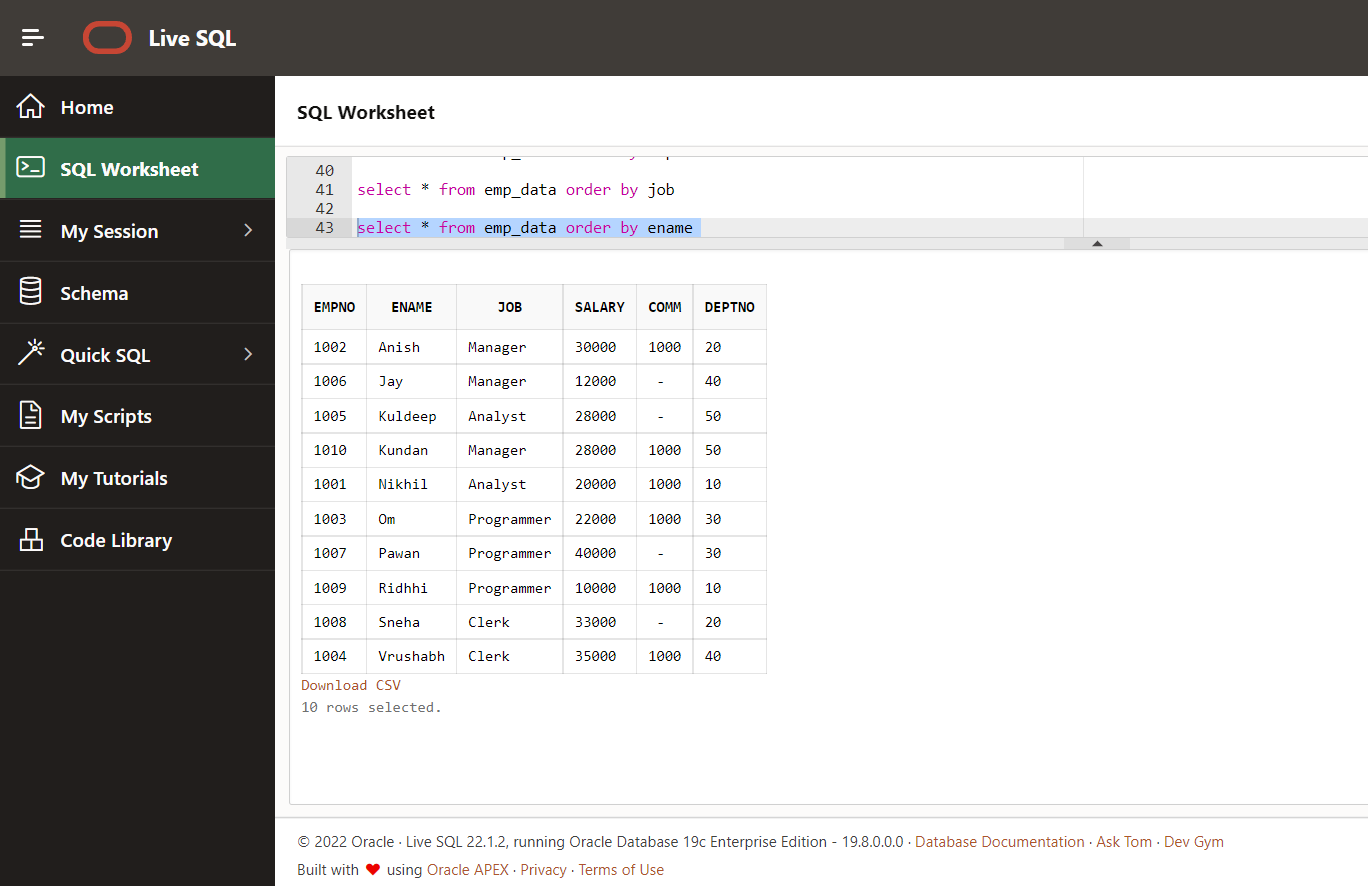
select \* from emp\_data order by job;

Output:



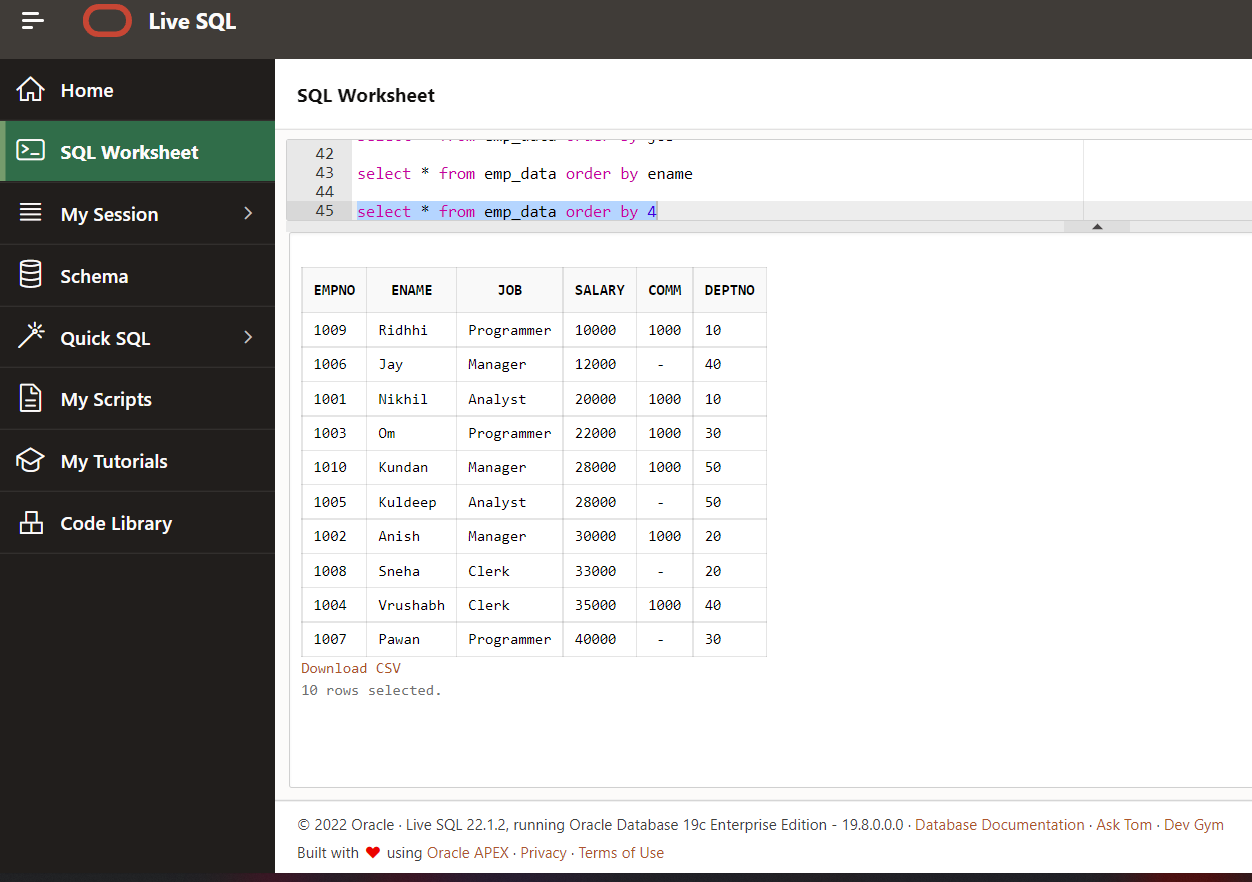
select \* from emp\_data order by ename

Output:



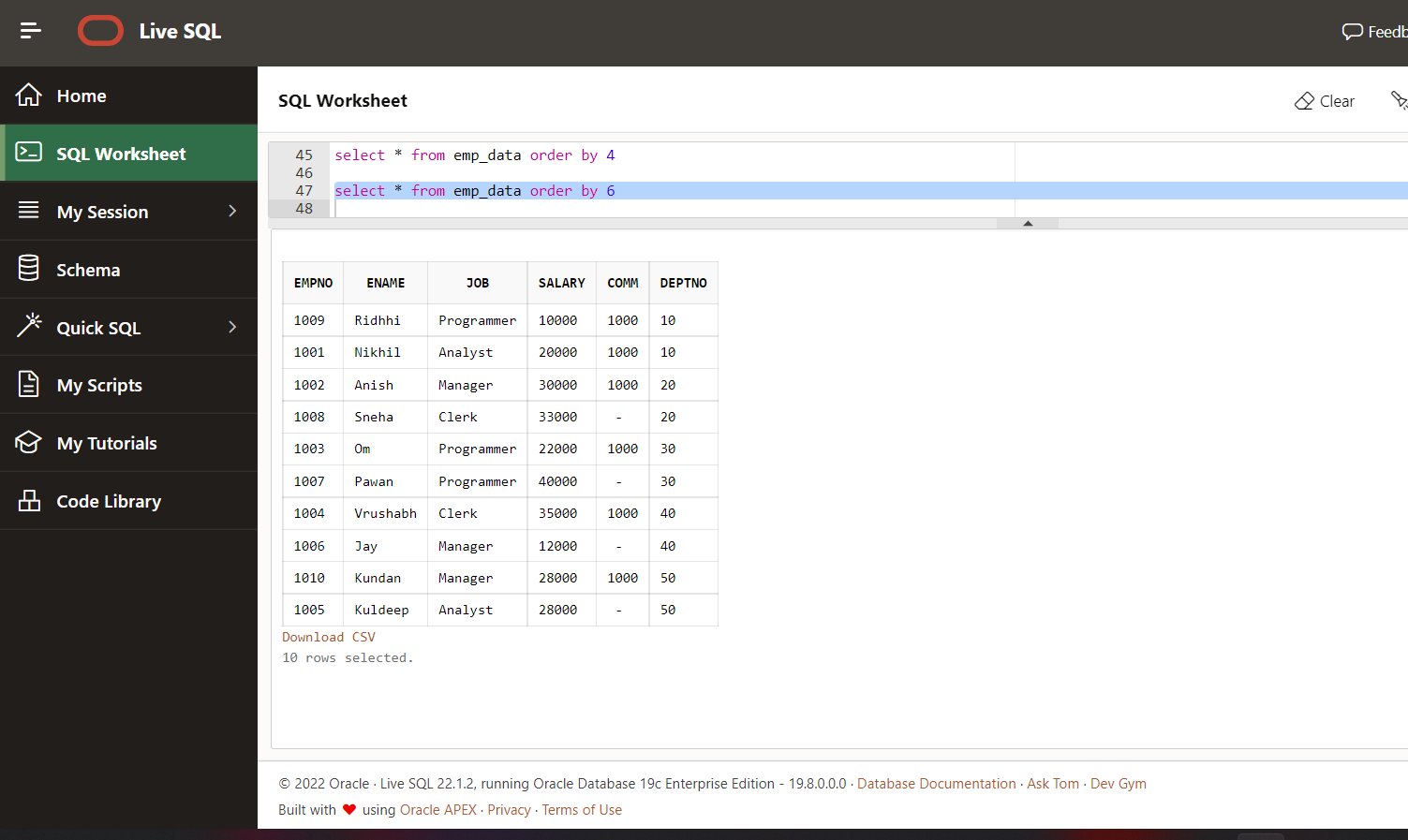
select \* from emp\_data order by 4

Output:



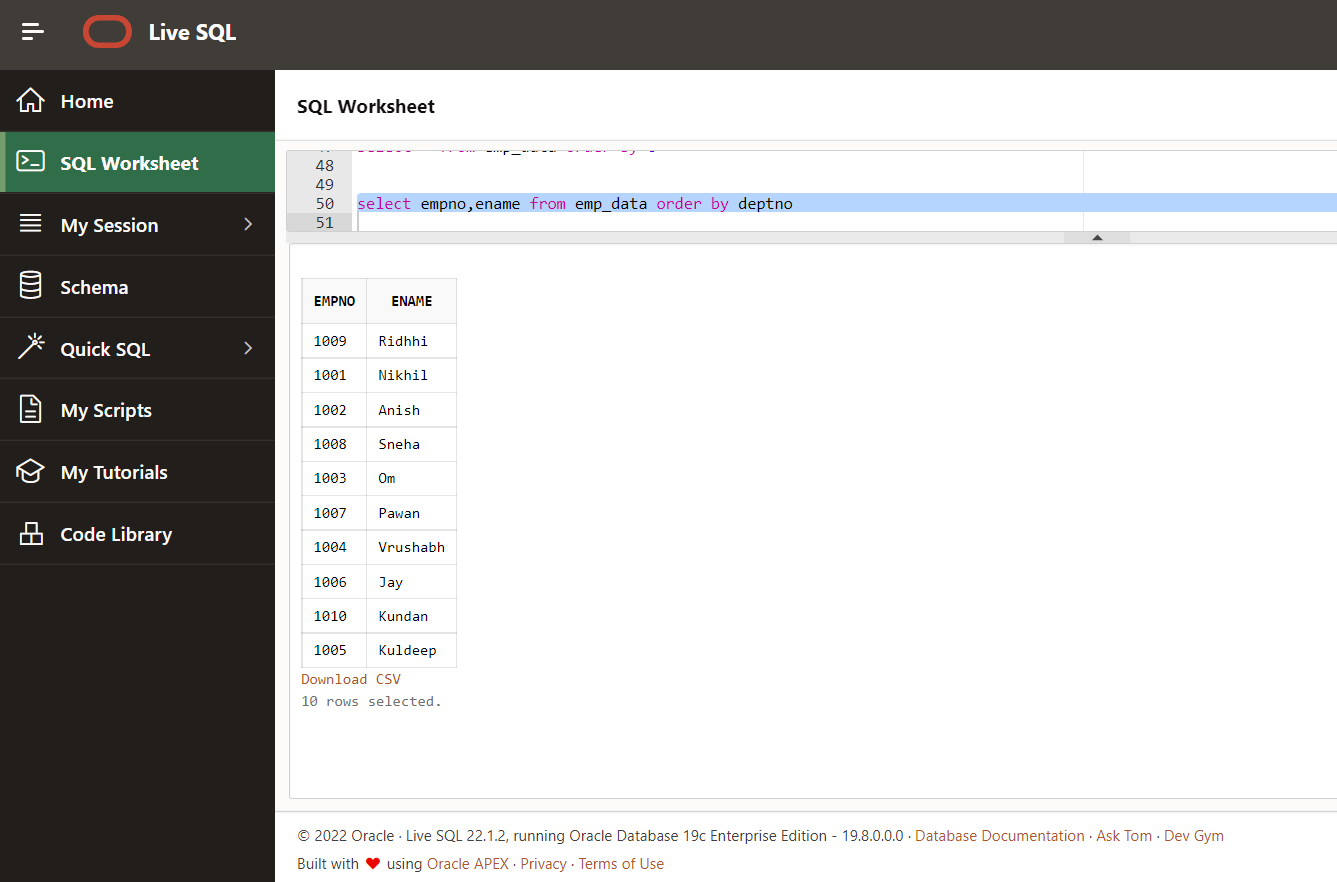
select \* from emp\_data order by 6

Output:



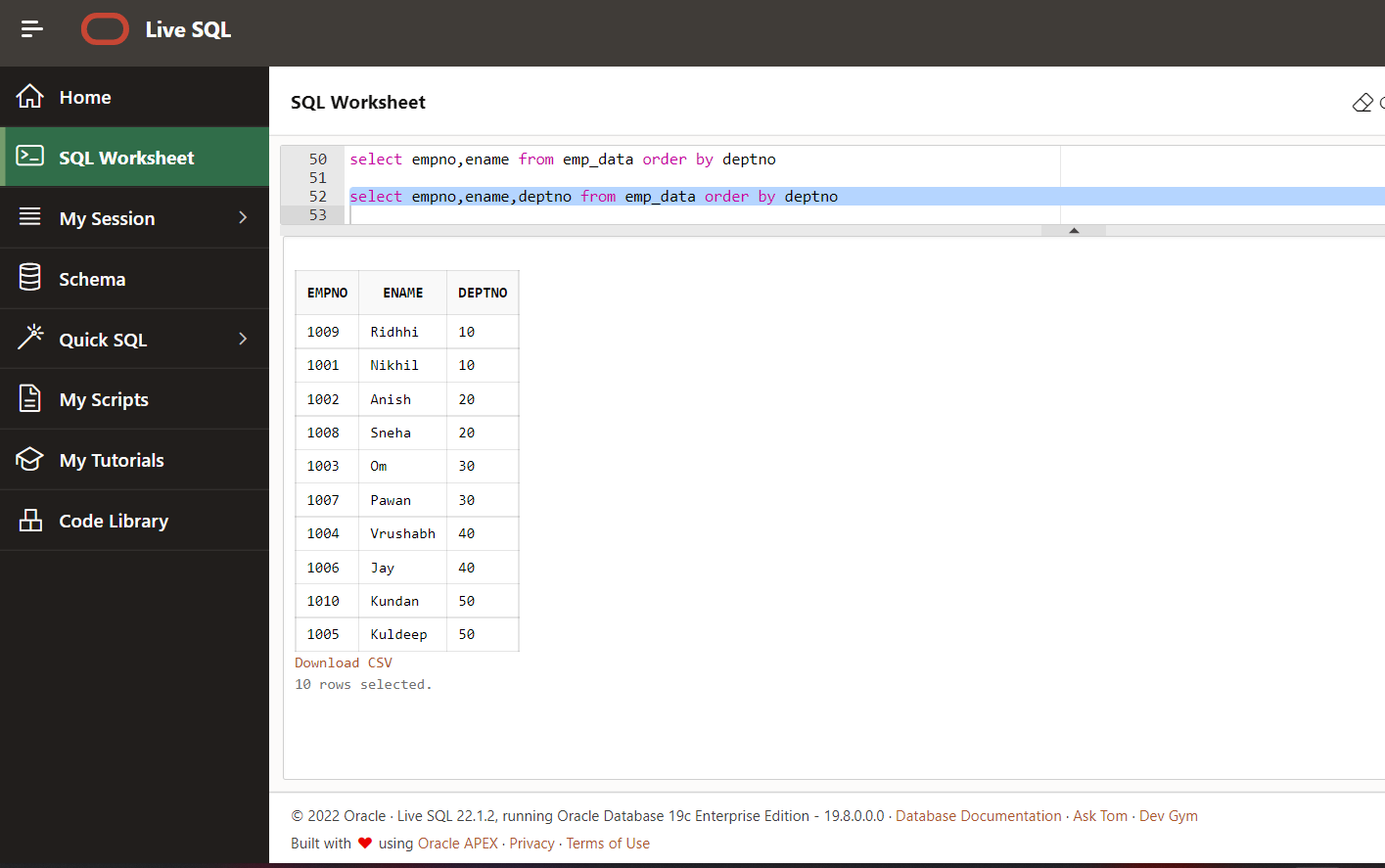
select empno,ename from emp\_data order by deptno

Output:



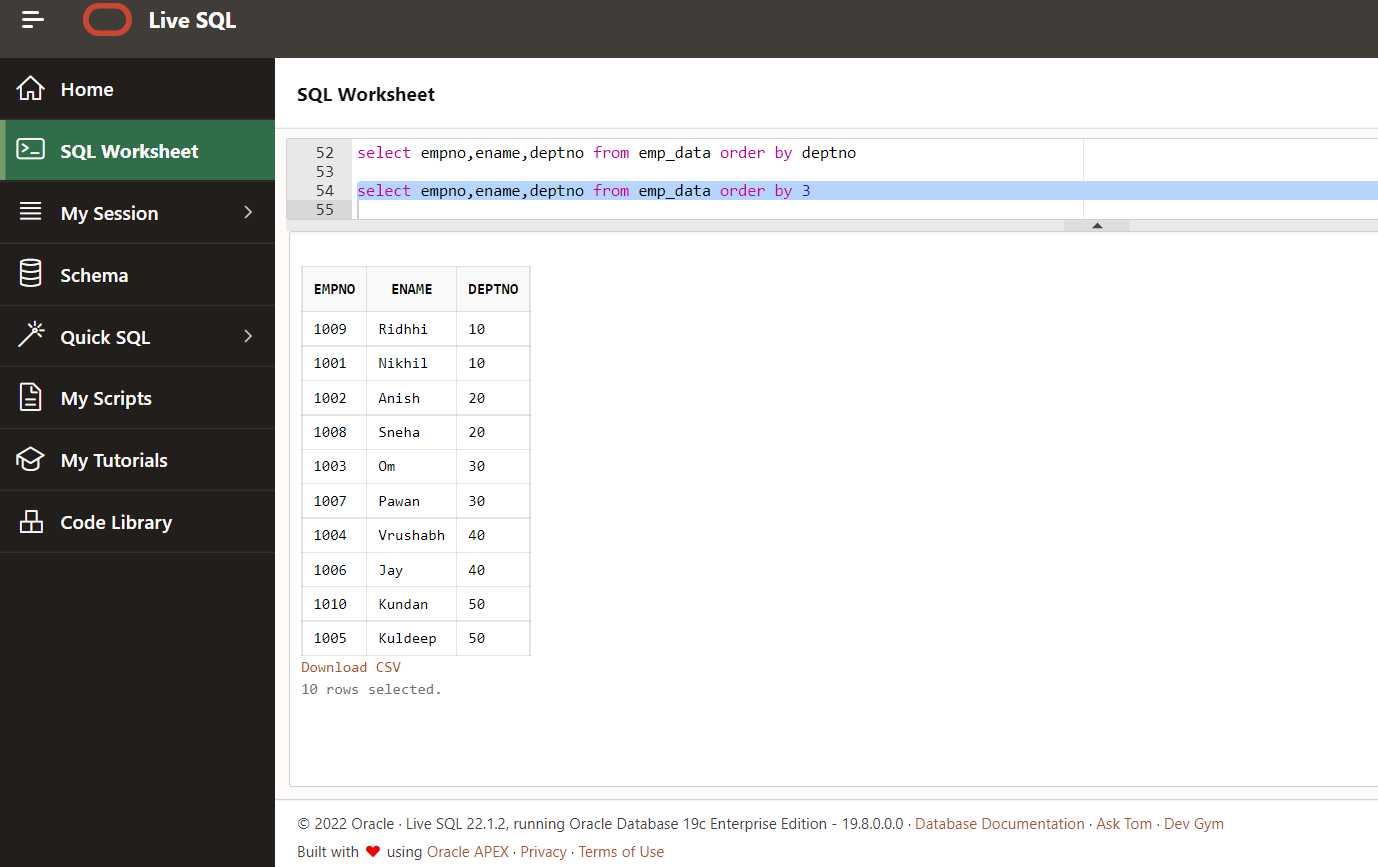
select empno,ename,deptno from emp\_data order by deptno

Output:



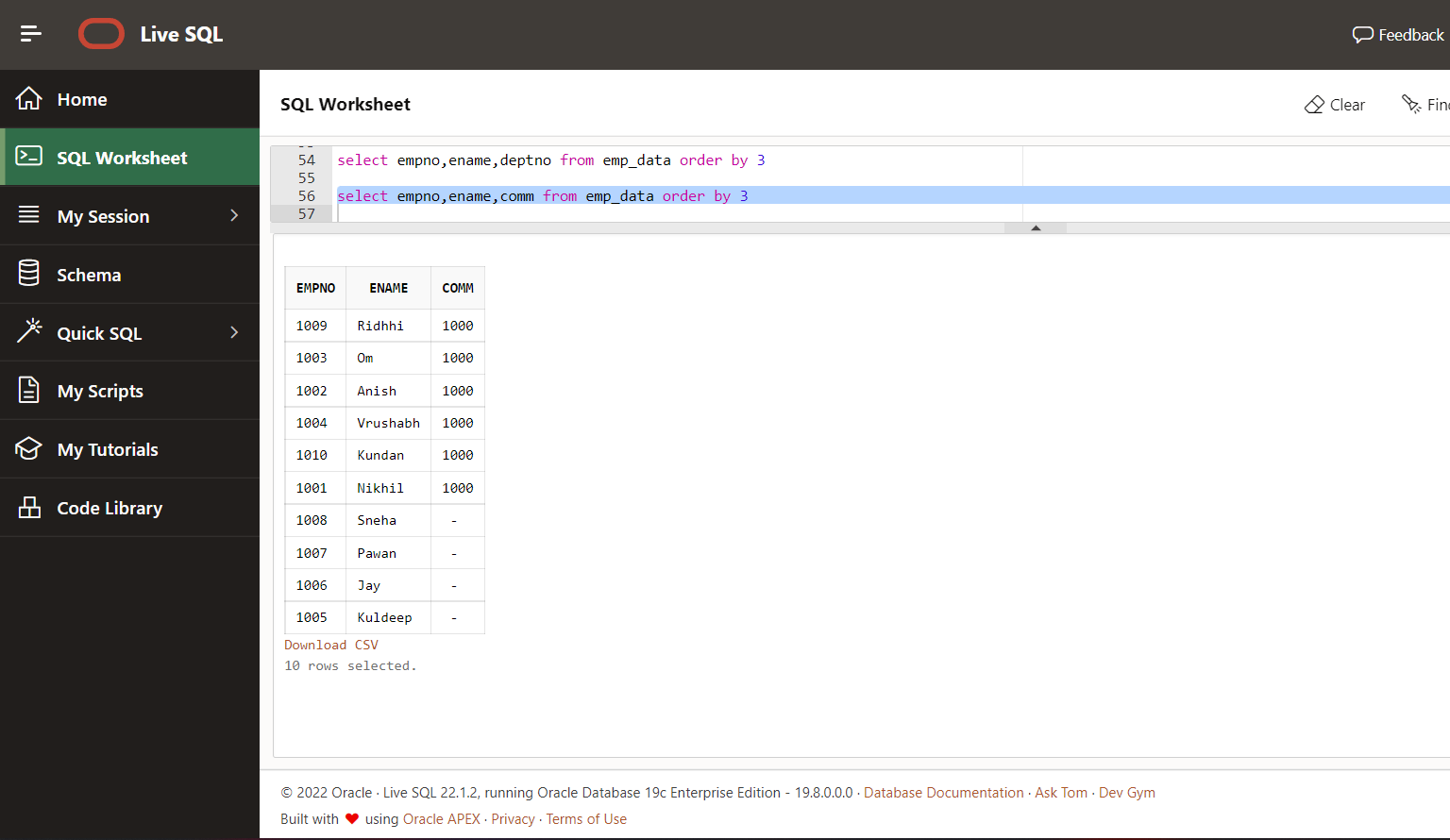
select empno,ename,deptno from emp\_data order by 3

Output:



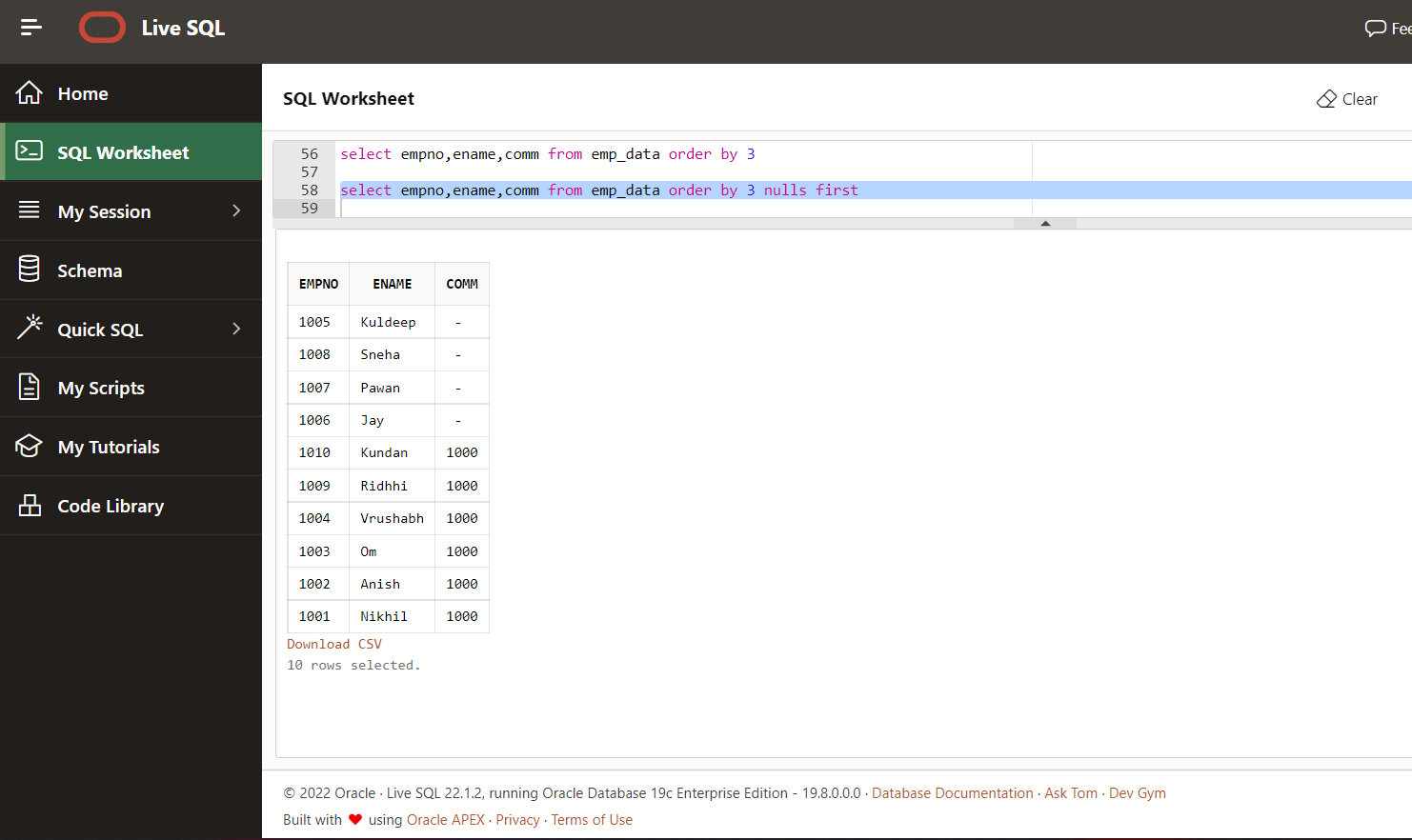
select empno,ename,comm from emp\_data order by 3

Output:



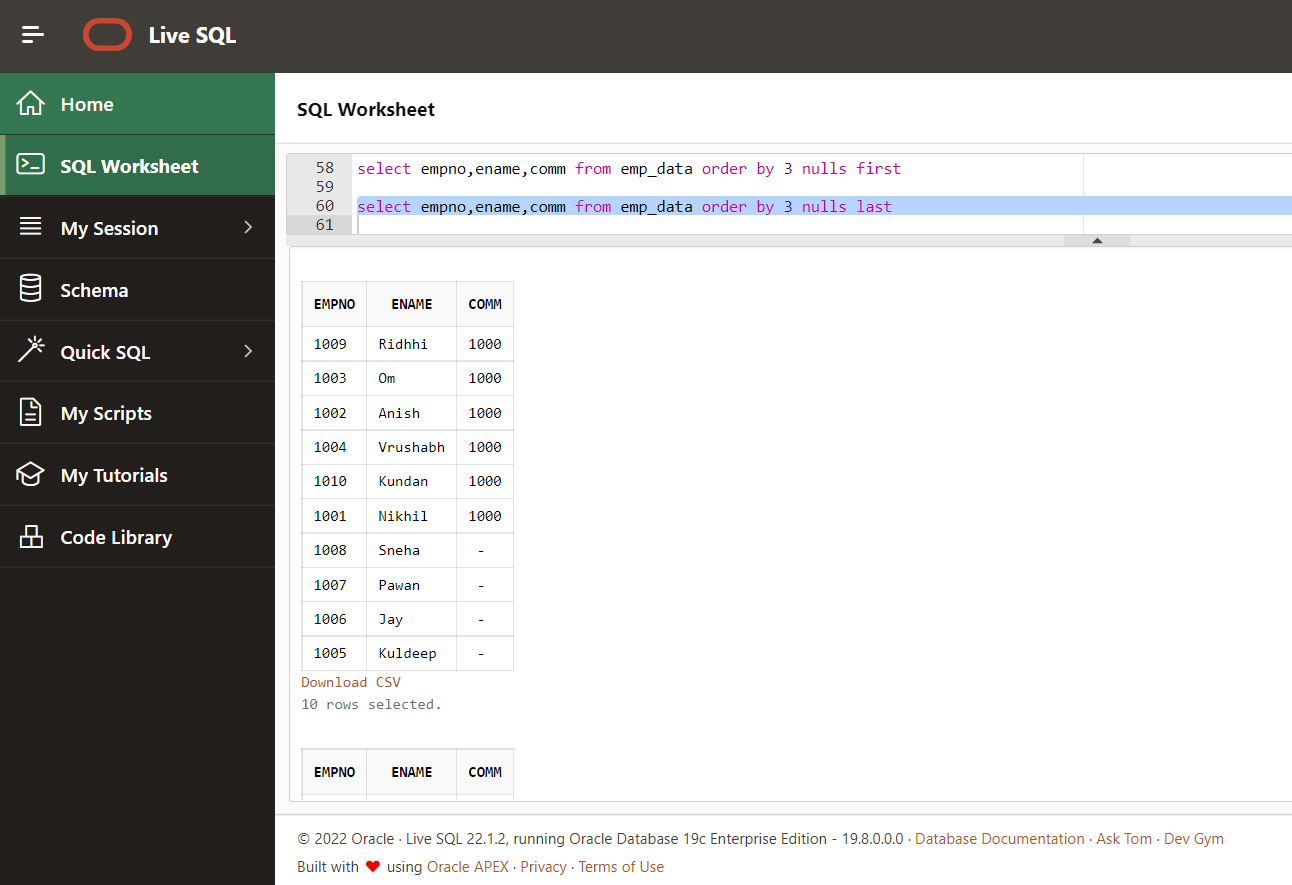
select empno,ename,comm from emp\_data order by 3 nulls first

Output:



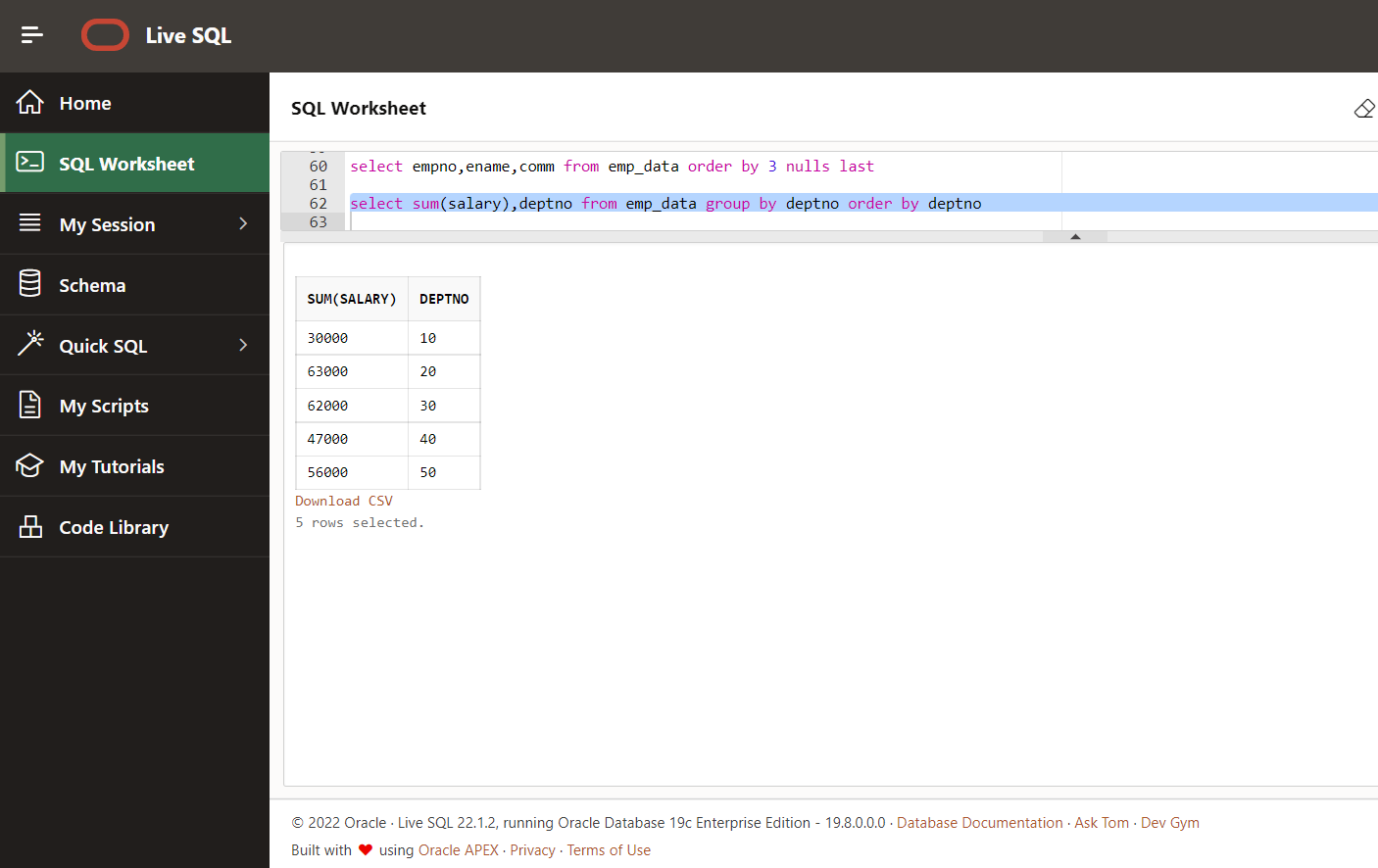
select empno,ename,comm from emp\_data order by 3 nulls last

Output:



select sum(salary),deptno from emp\_data group by deptno order by deptno

Output:



select \* from emp\_data

Output:

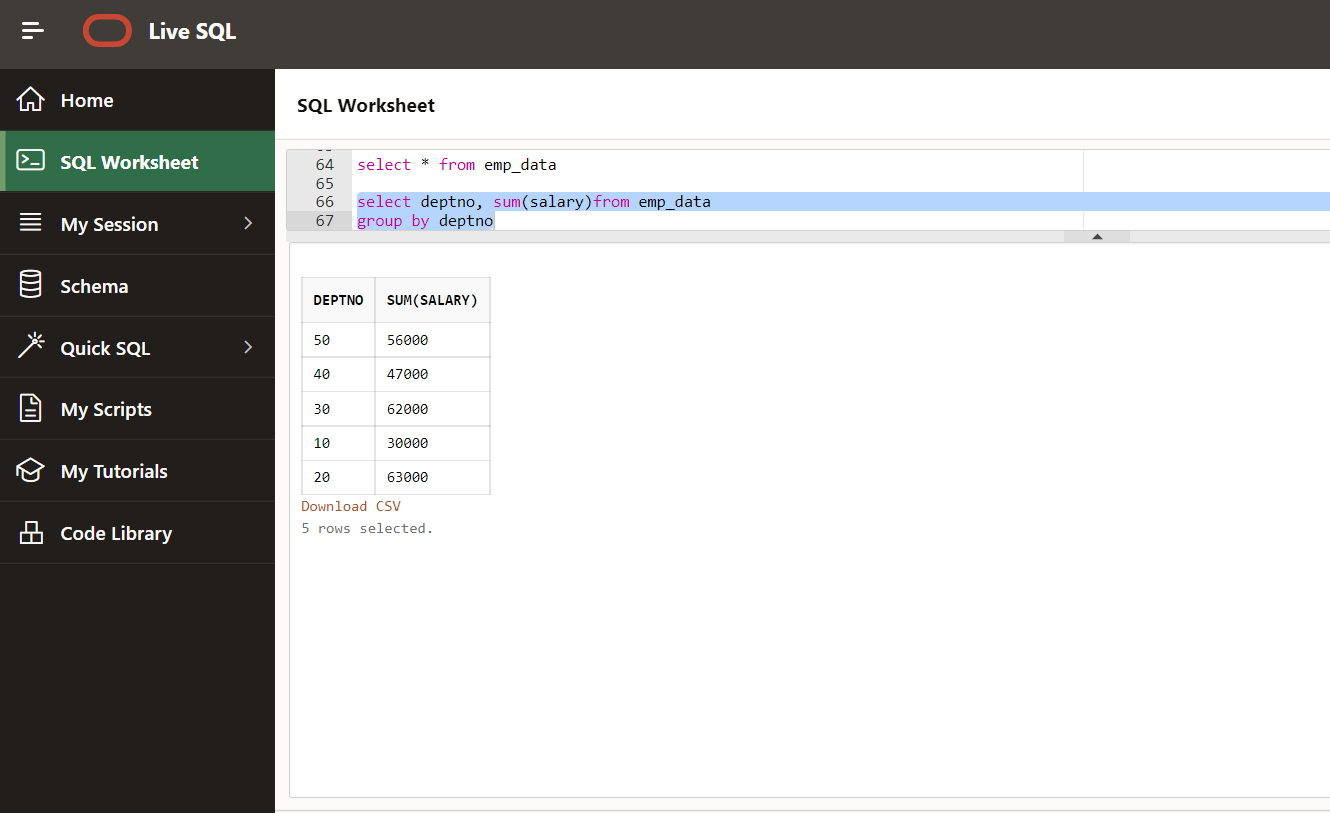


Group By

select deptno, sum(salary)from emp\_data

group by deptno

Output:



select deptno, sum(salary)from emp\_data

group by deptno

order by deptno

Output:



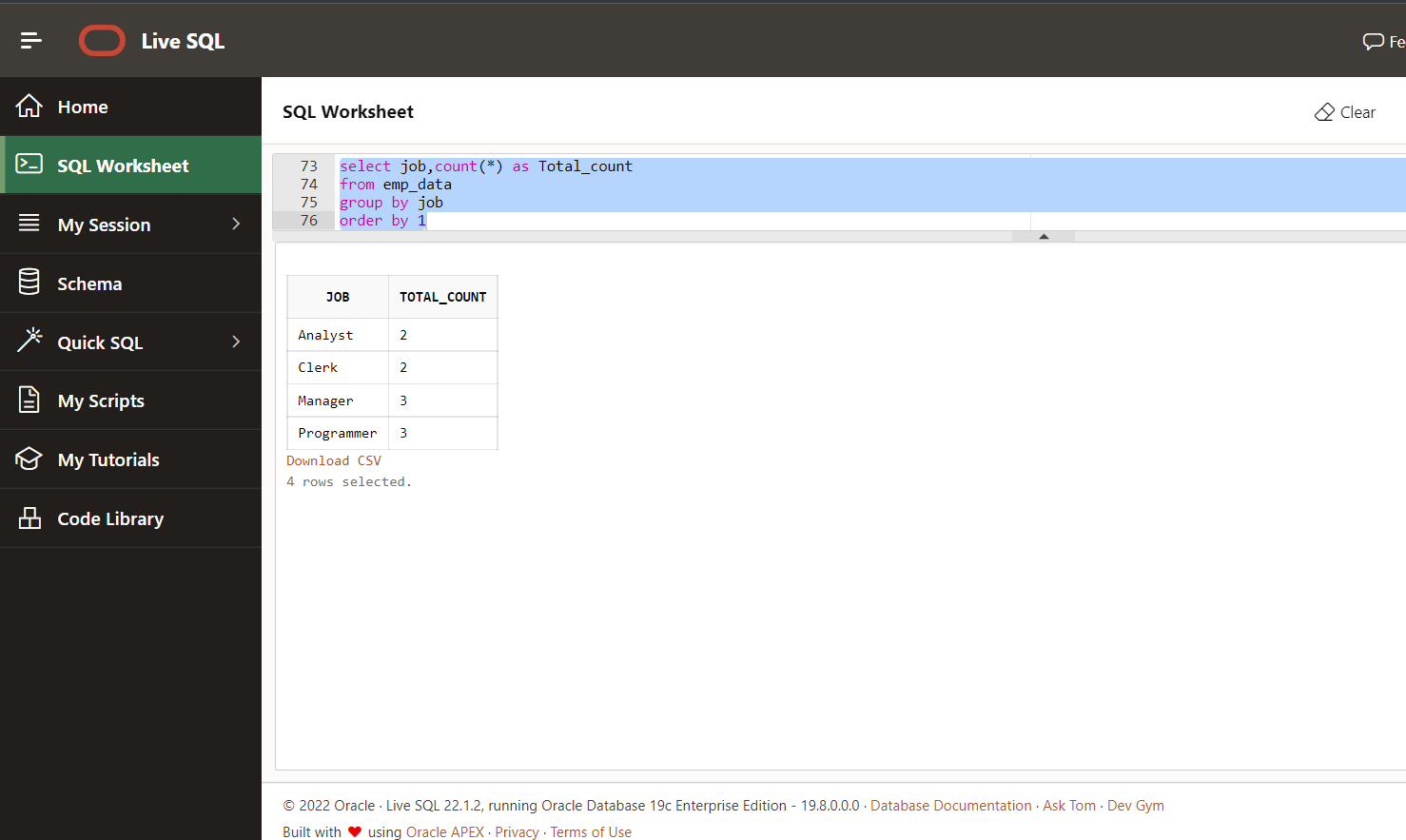
select job,count(\*) as Total\_count

from emp\_data

group by job

order by 1

Output:



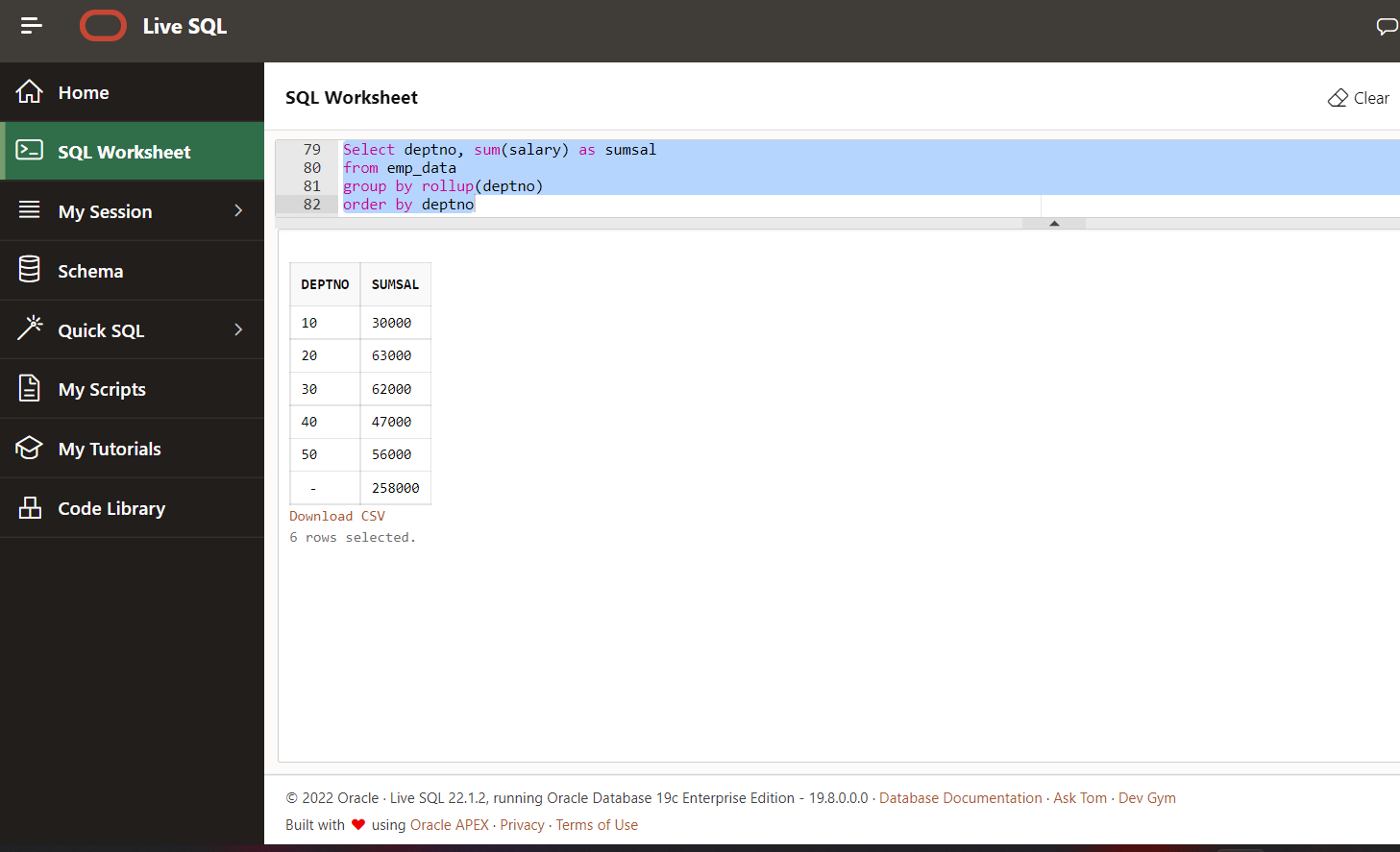
Select deptno, sum(salary) as sumsal

from emp\_data

 group by rollup(deptno)

order by dept no

Output:



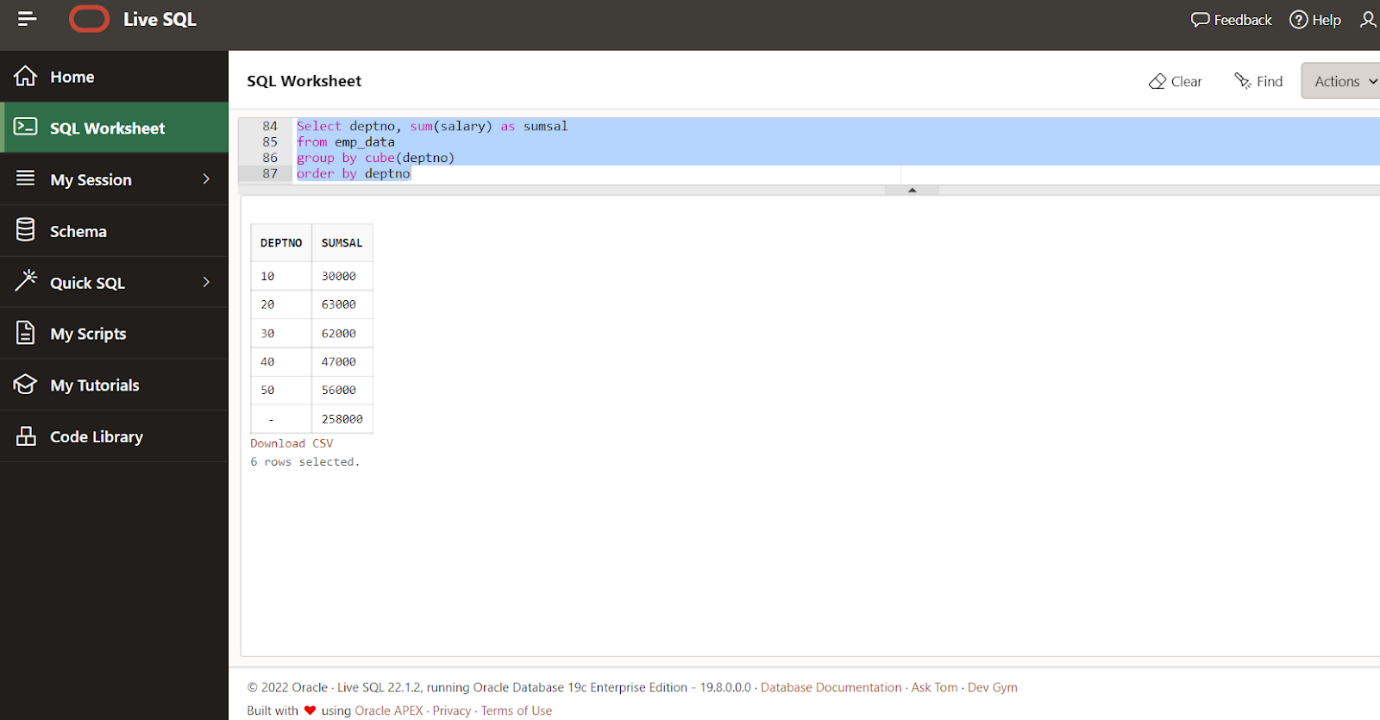
Select deptno, sum(salary) as sum sal

from emp\_data

 group by cube(deptno)

order by dept no

Output:



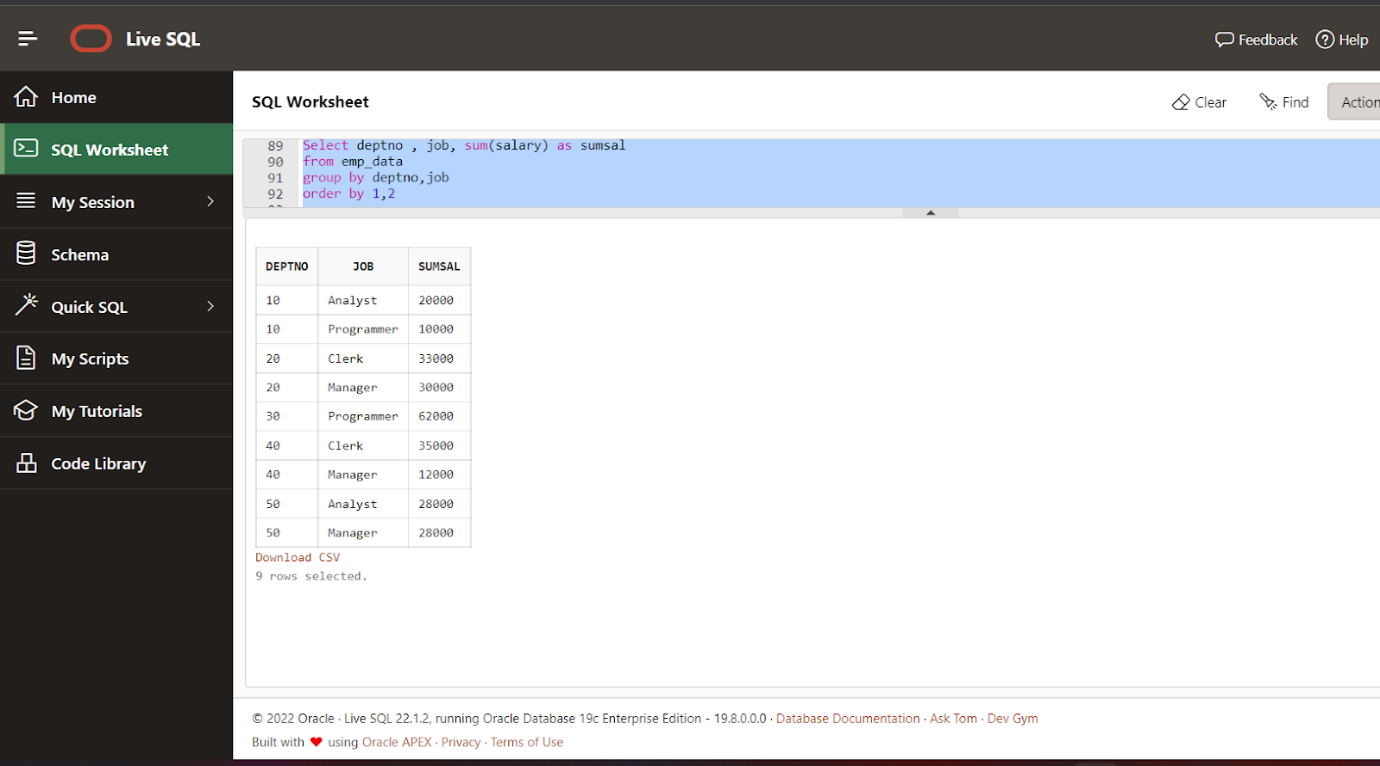
Select deptno , job, sum(salary) as sumsal

from emp\_data

group by deptno,job

order by 1,2

Output:



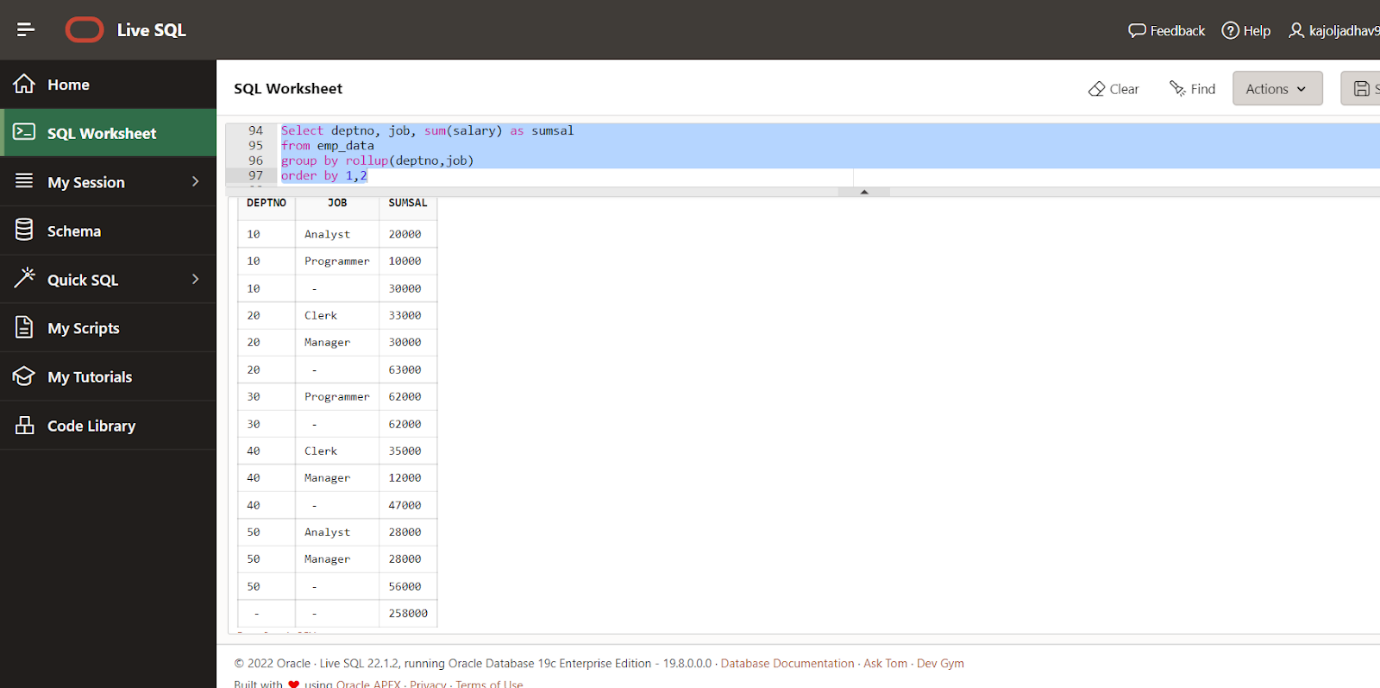
Select deptno, job, sum(salary) as sumsal

from emp\_data

group by rollup(deptno,job)

order by 1,2

Output:



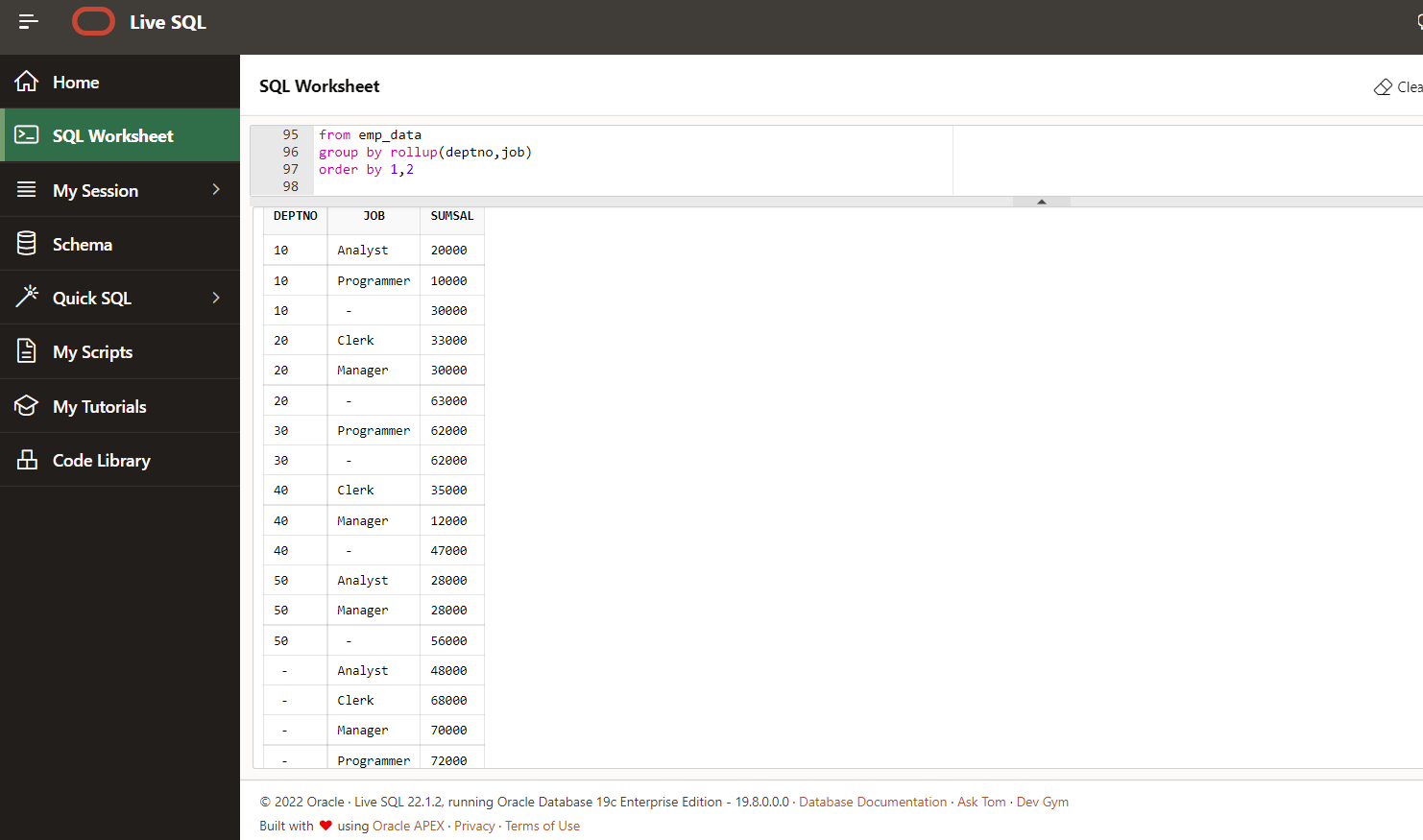
Select deptno, job, sum(salary) as sumsal

from emp\_data

group by cube(deptno,job)

order by 1,2

Output:



Select deptno, job, sum(salary) as sumsal,

case grouping\_id(deptno,job)

when 1 then ‘Dept Subtotal’

when 2 then ‘Job Subtotal’

when 3 then ‘Grand Total’

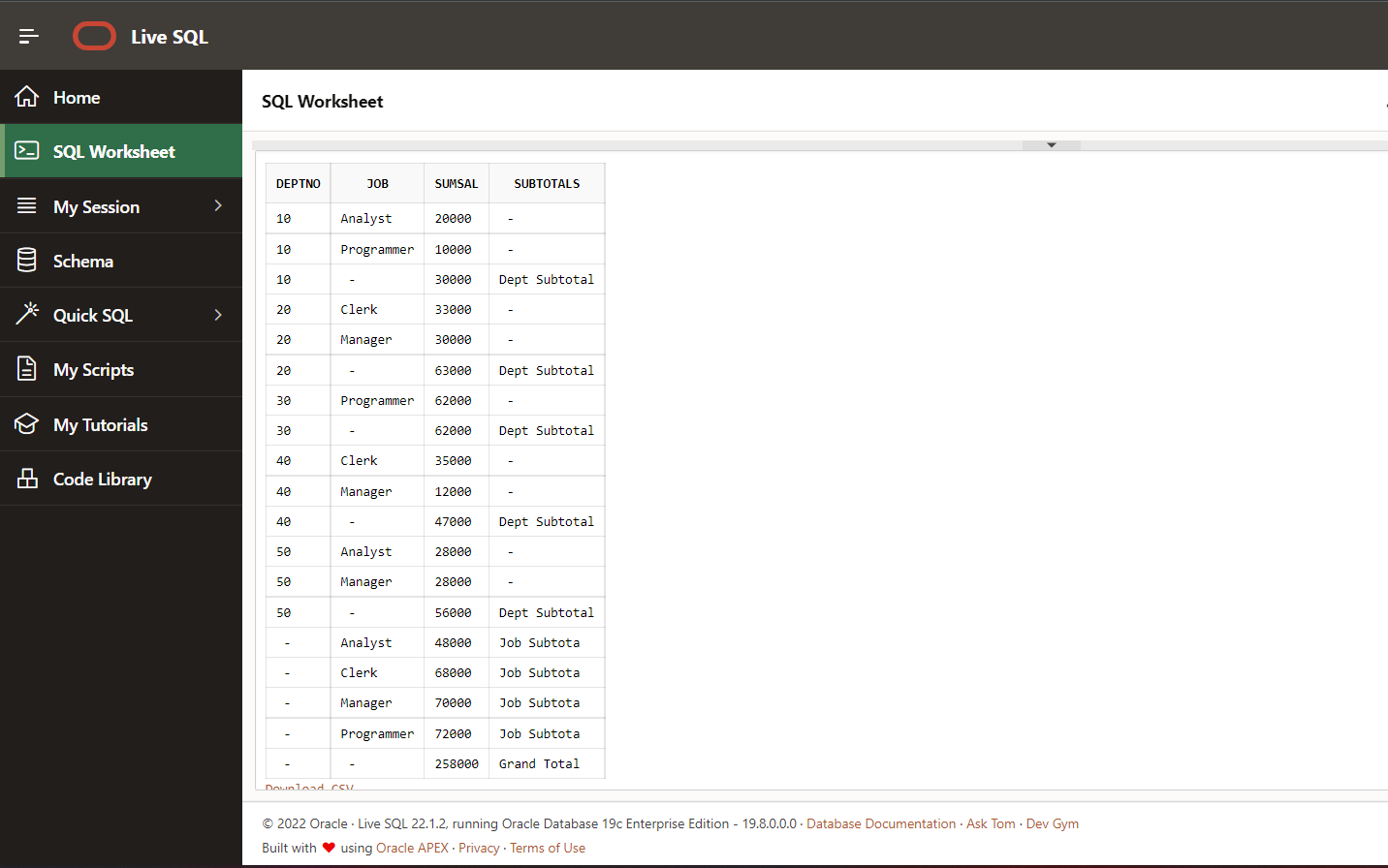
end as subtotals

from emp\_data

group by cube(deptno,job)

order by 1,2

Output:



RANKING:

**TO ASSIGN RANK NUMBERS TO EACH ROW OR ELSE GROUP OF ROWS WISE**

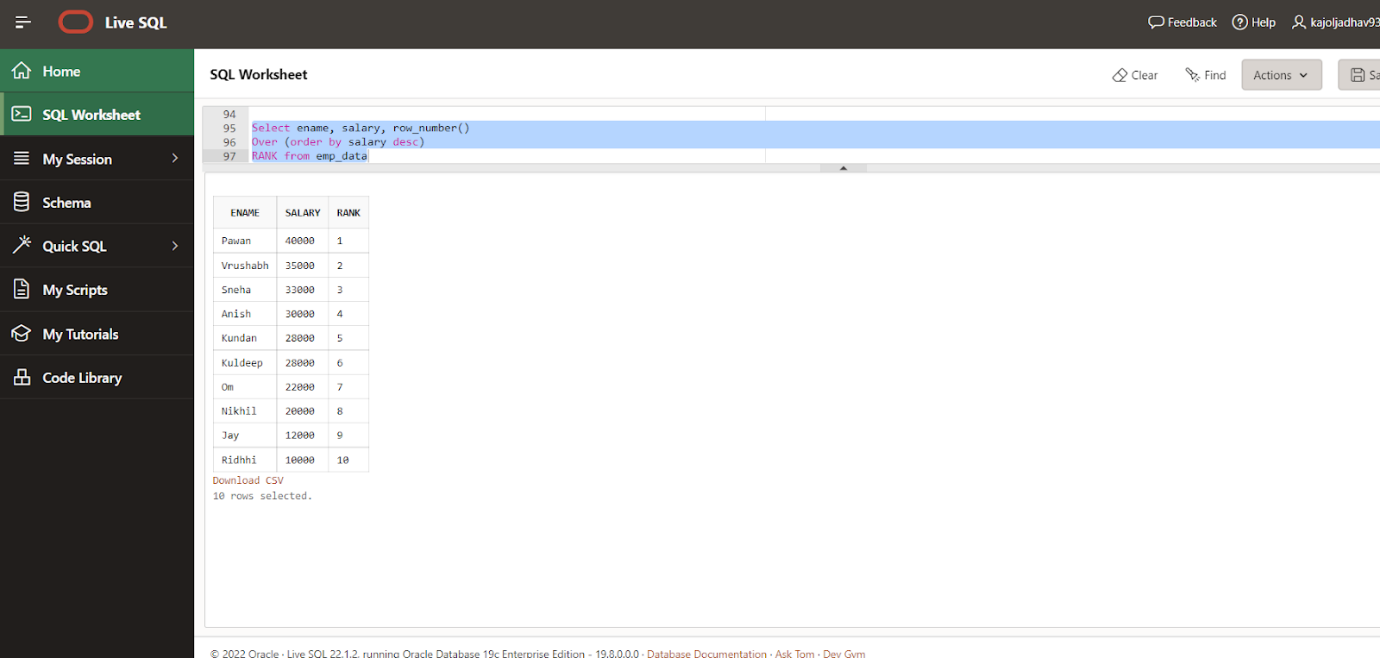
Code:

Select ename, salary, row\_number()

Over (order by salary)

RANK from emp\_data

Output:

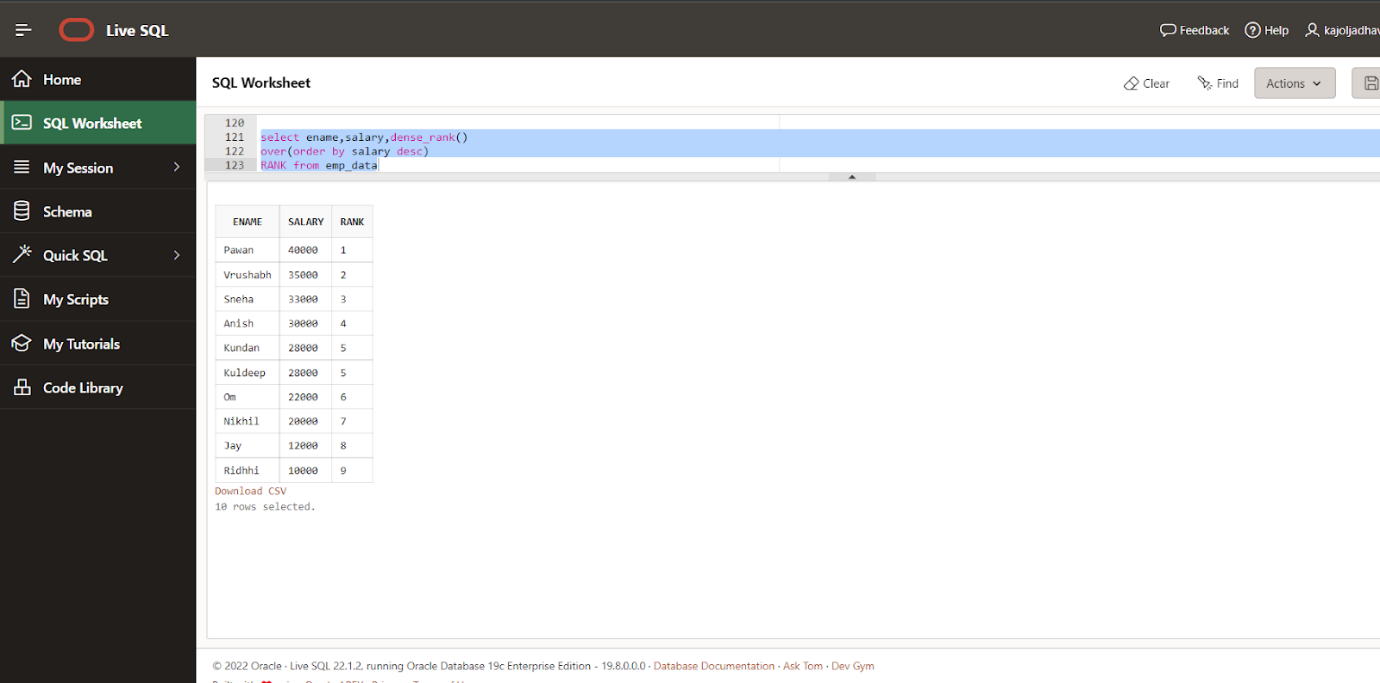


select ename,salary,dense\_rank()

over(order by salary desc)

RANK from emp\_data

Output:

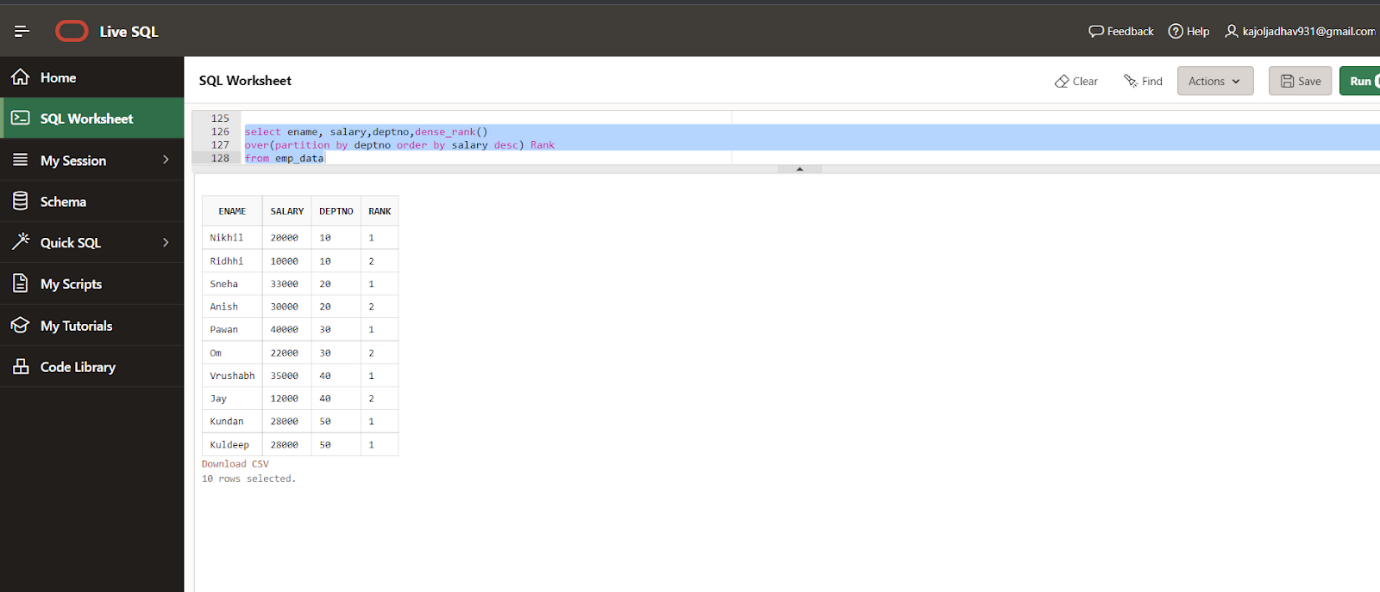


select ename, salary,deptno,dense\_rank()

over(partition by deptno order by salary desc) Rank

from emp\_data

Output:



Lead and Lag:-

create table company

(

yearname number,

prod\_sale number,

sale\_value number

)

insert into company values(2015,100,2500);

insert into company values(2016,140,10000);

insert into company values(2017,300,20000);

insert into company values(2018,80,15000);

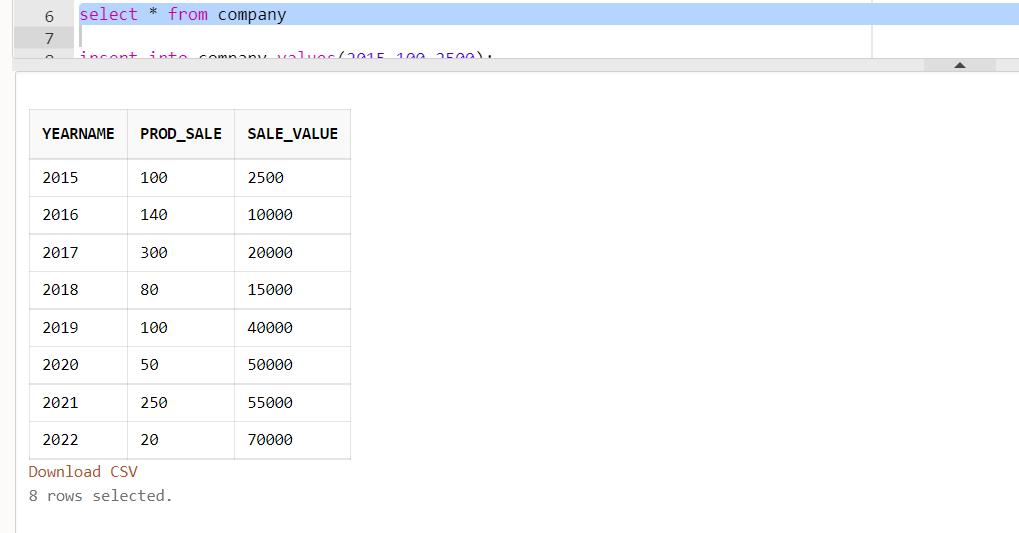
insert into company values(2019,100,40000);

insert into company values(2020,50,50000);

insert into company values(2021,250,55000);

insert into company values(2022,20,70000);

select \* from company;

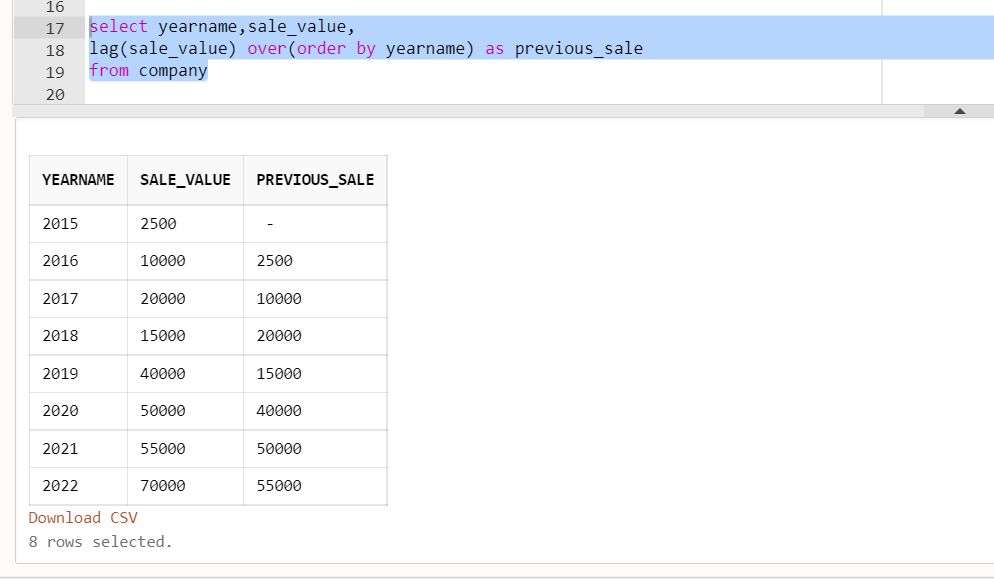


Query:-

select yearname,sale\_value,

lag(sale\_value) over(order by yearname) as previous\_sale

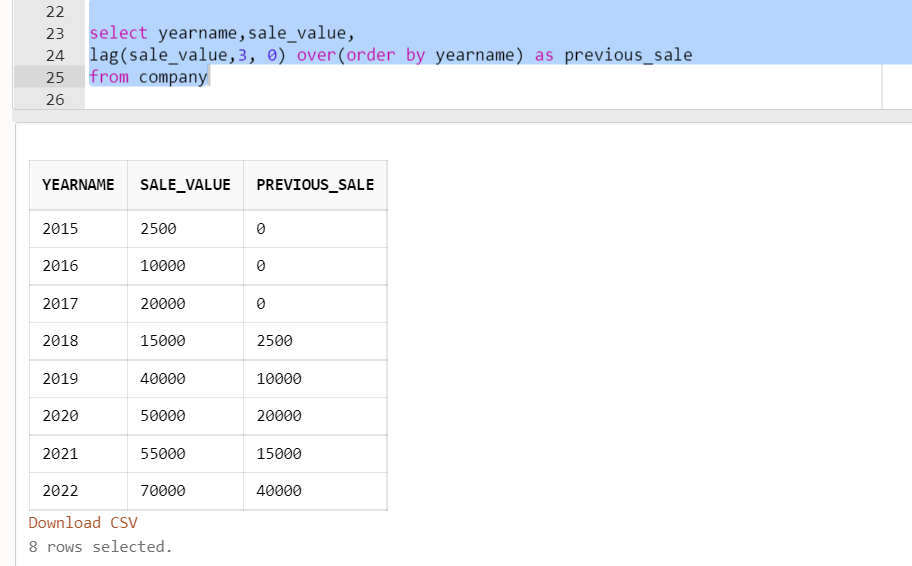
from company



select yearname,sale\_value,

lag(sale\_value,3, 0) over(order by yearname) as previous\_sale

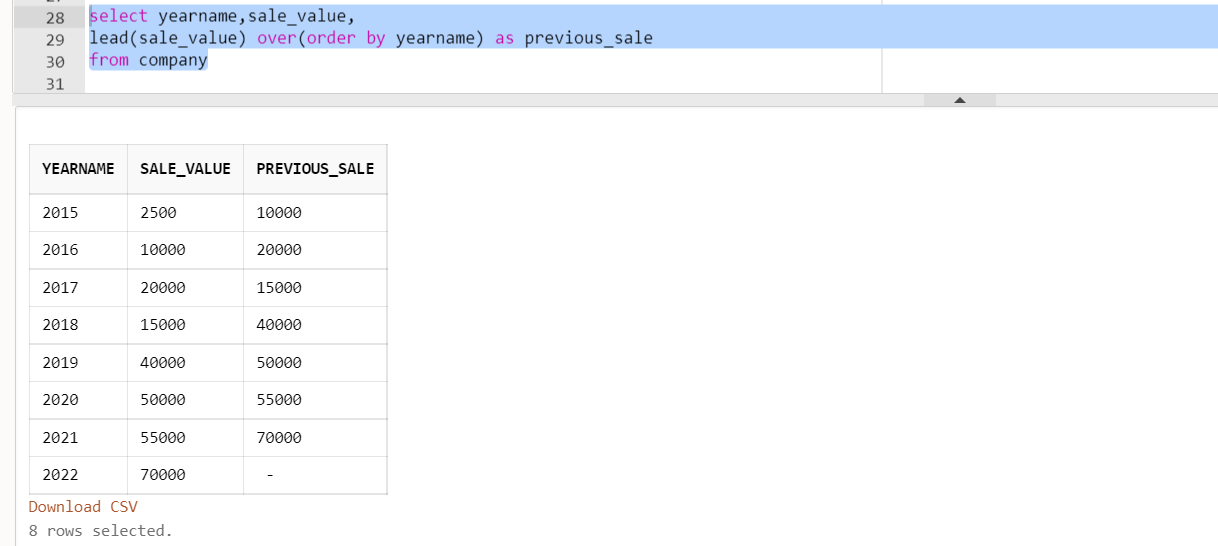
from company



select yearname,sale\_value,

lead(sale\_value) over(order by yearname) as previous\_sale

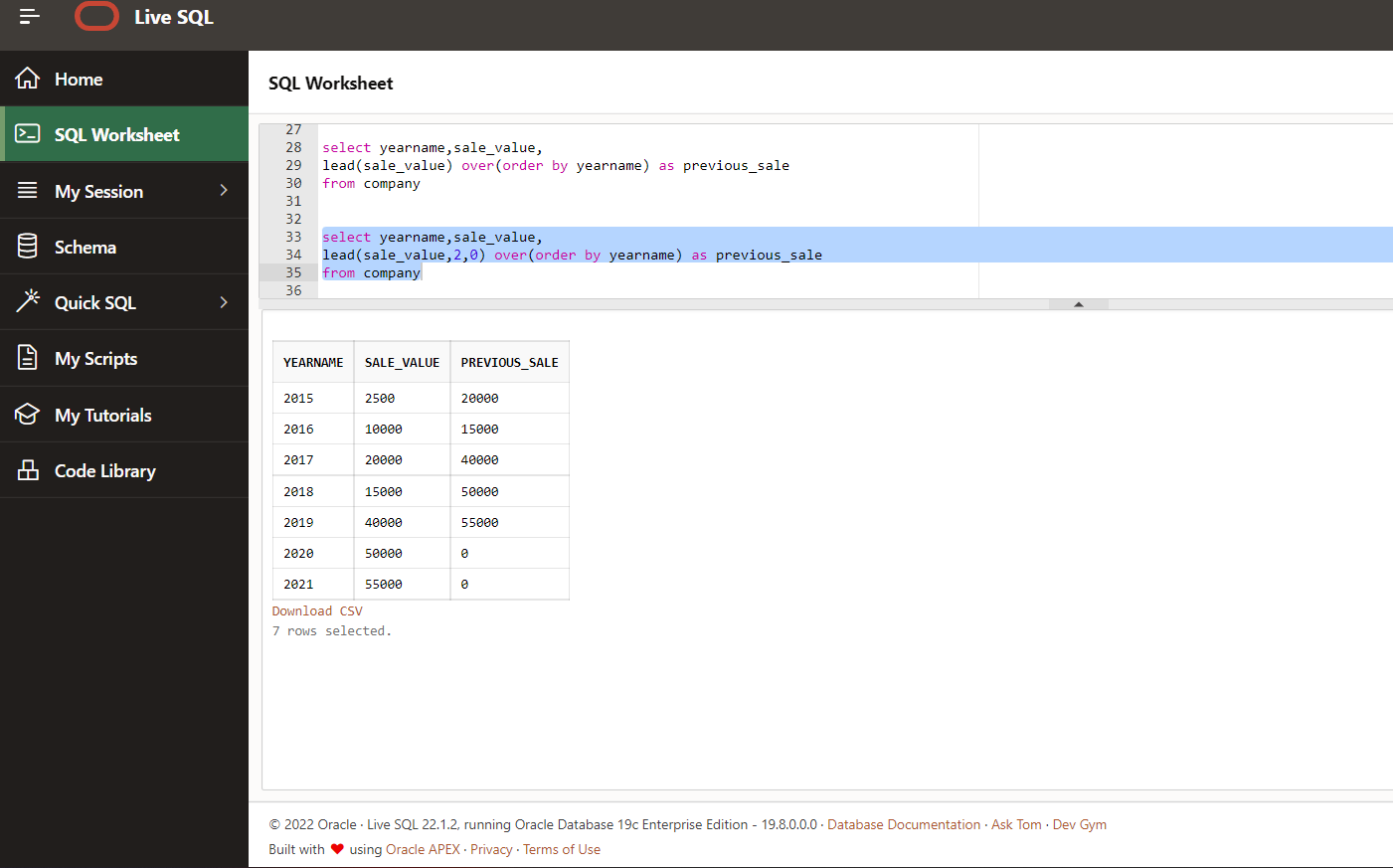
from company



select yearname,sale\_value,

lead(sale\_value,2,0) over(order by yearname) as previous\_sale

from company



**Using lag/lead multiples times in one query**

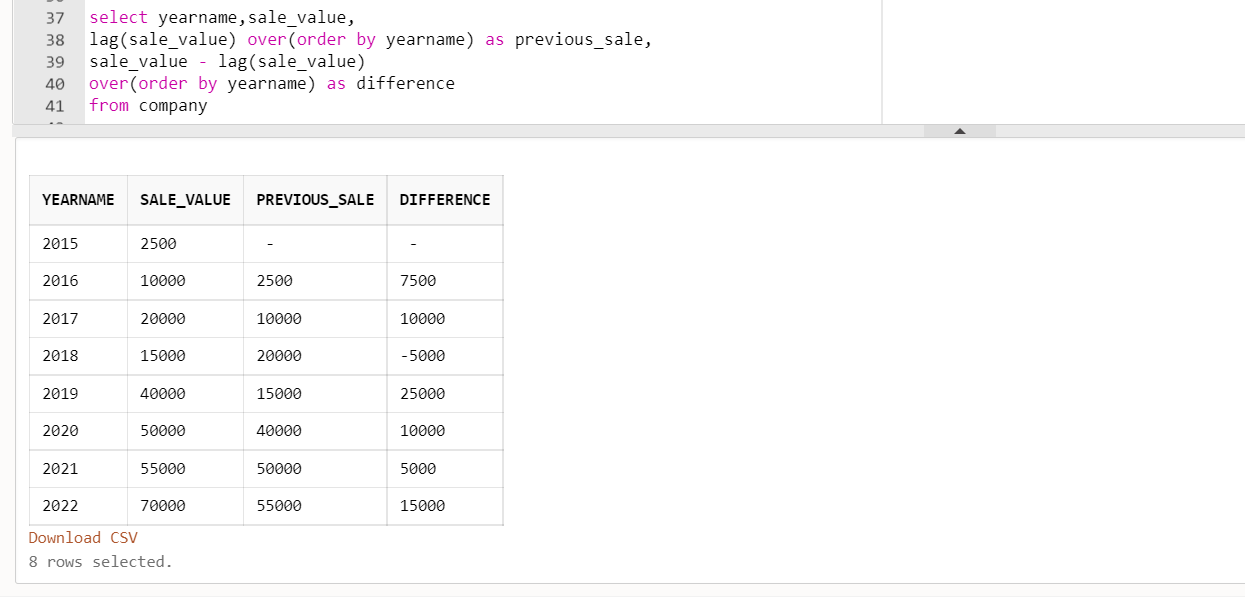
select yearname,sale\_value,

lag(sale\_value) over(order by yearname) as previous\_sale,

sale\_value - lag(sale\_value)

over(order by yearname) as difference

from company



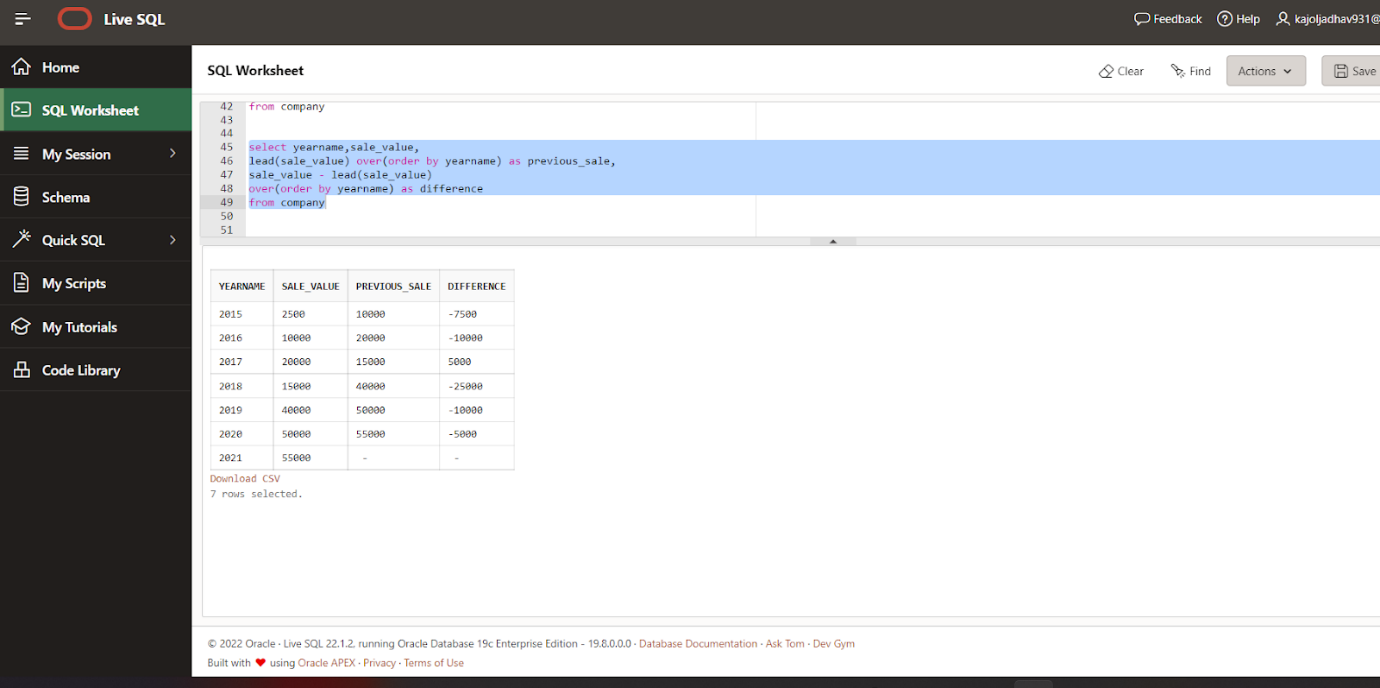
select yearname,sale\_value,

lead(sale\_value) over(order by yearname) as previous\_sale,

sale\_value - lead(sale\_value)

over(order by yearname) as difference

from company



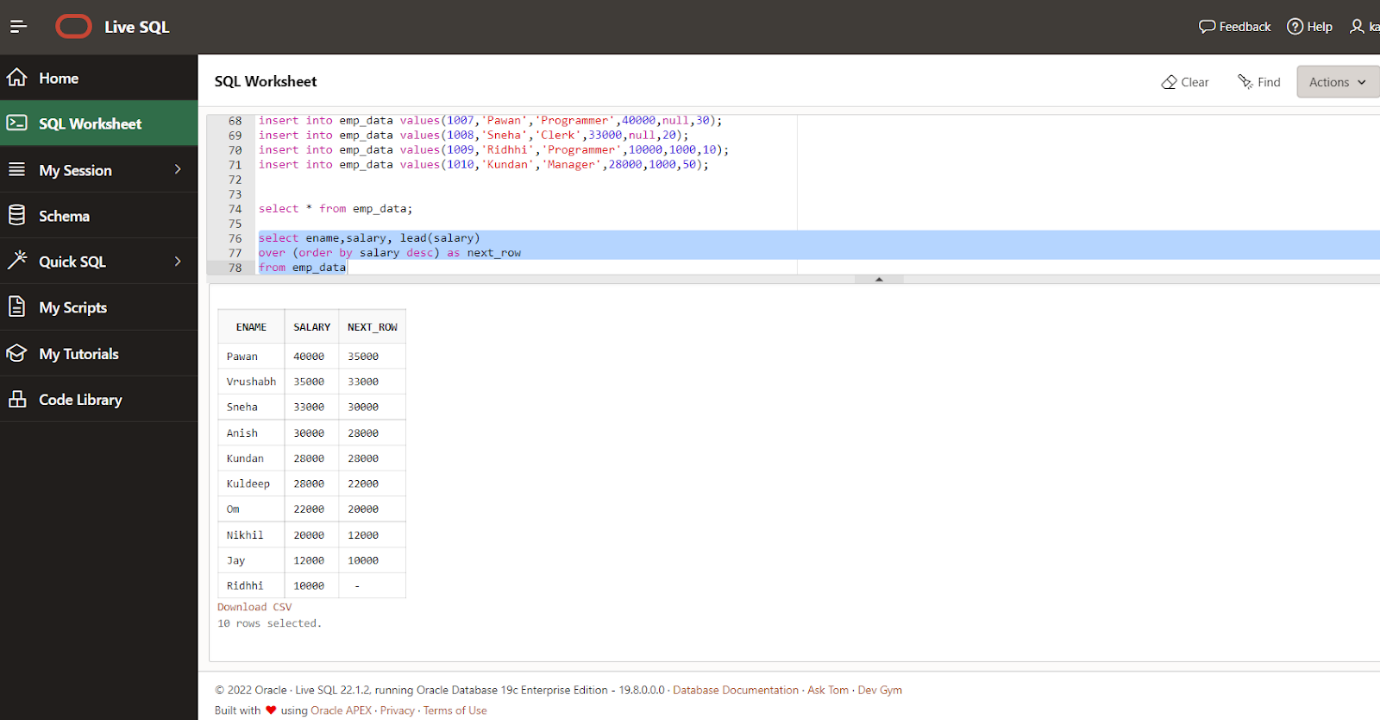
Select \* from emp\_data



select ename,salary, lead(salary)

over (order by salary desc) as next\_row

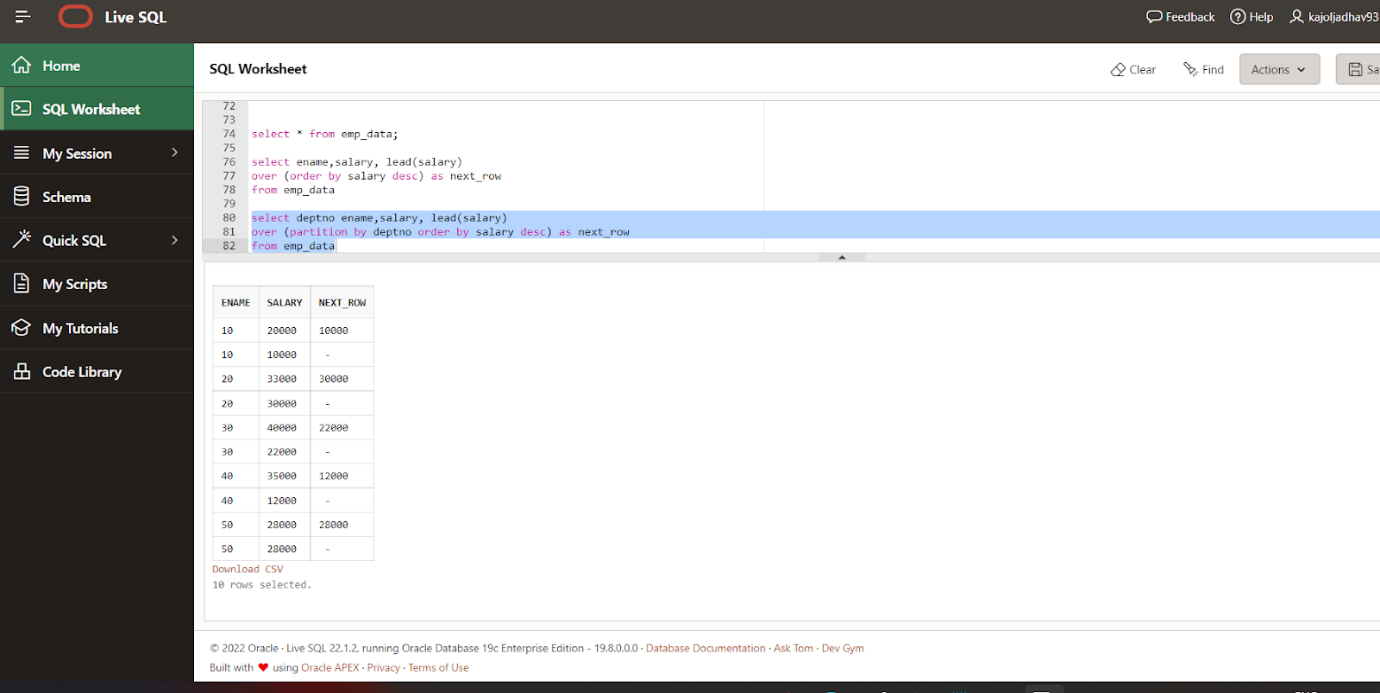
from emp\_data



select deptno ename,salary, lead(salary)

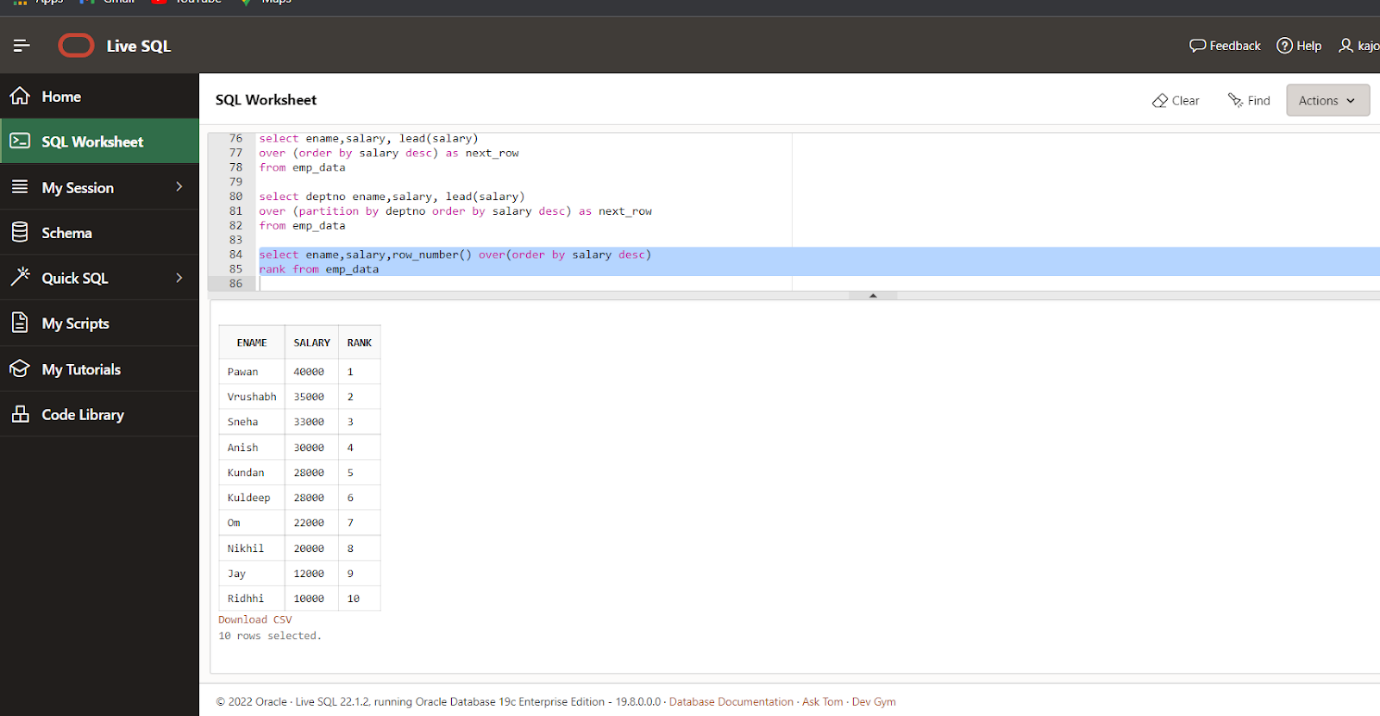
over (partition by deptno order by salary desc) as next\_row

from emp\_data



select ename,salary,row\_number() over(order by salary desc)

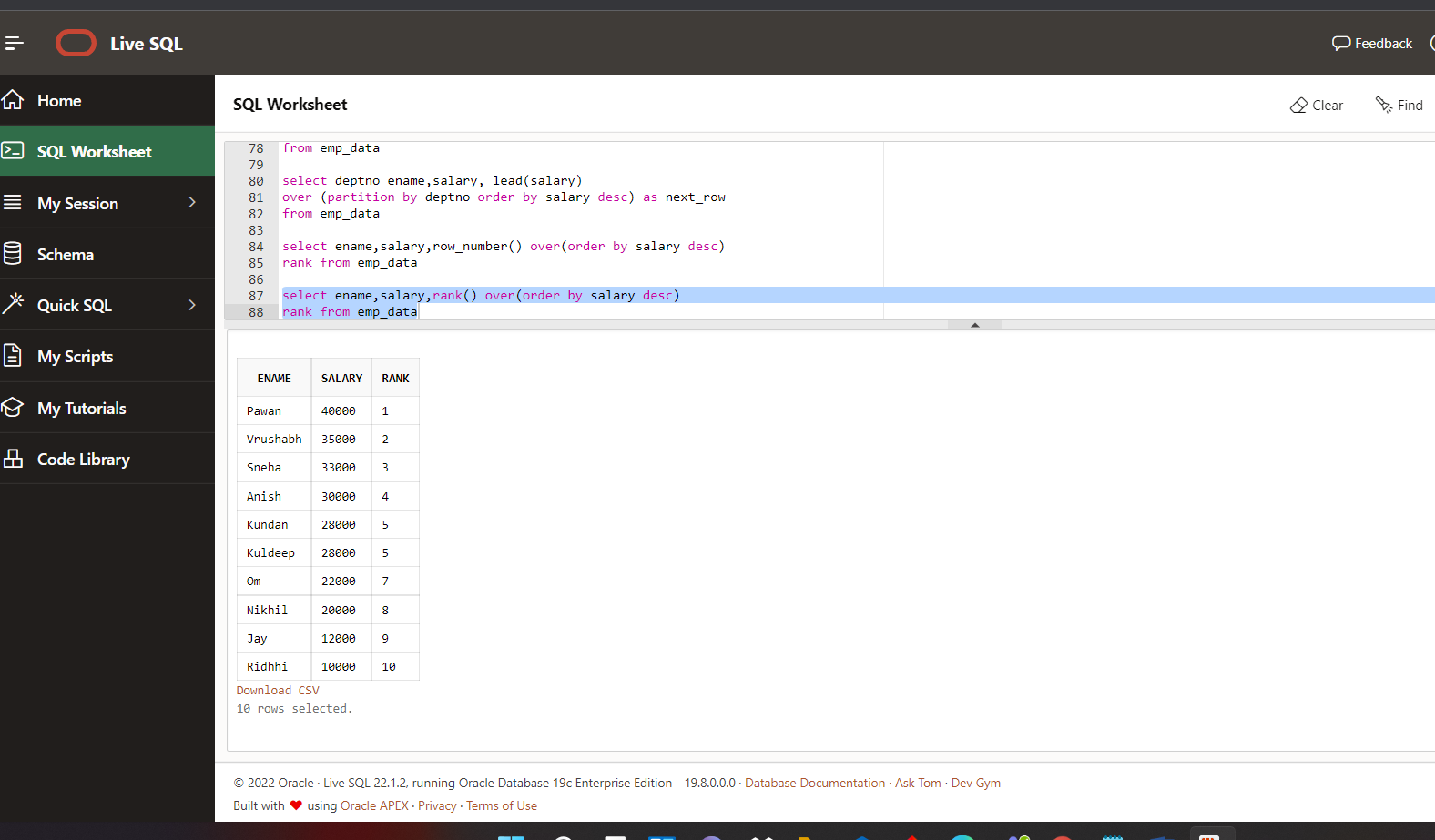
rank from emp\_data



Using rank

select ename,salary,rank() over(order by salary desc)

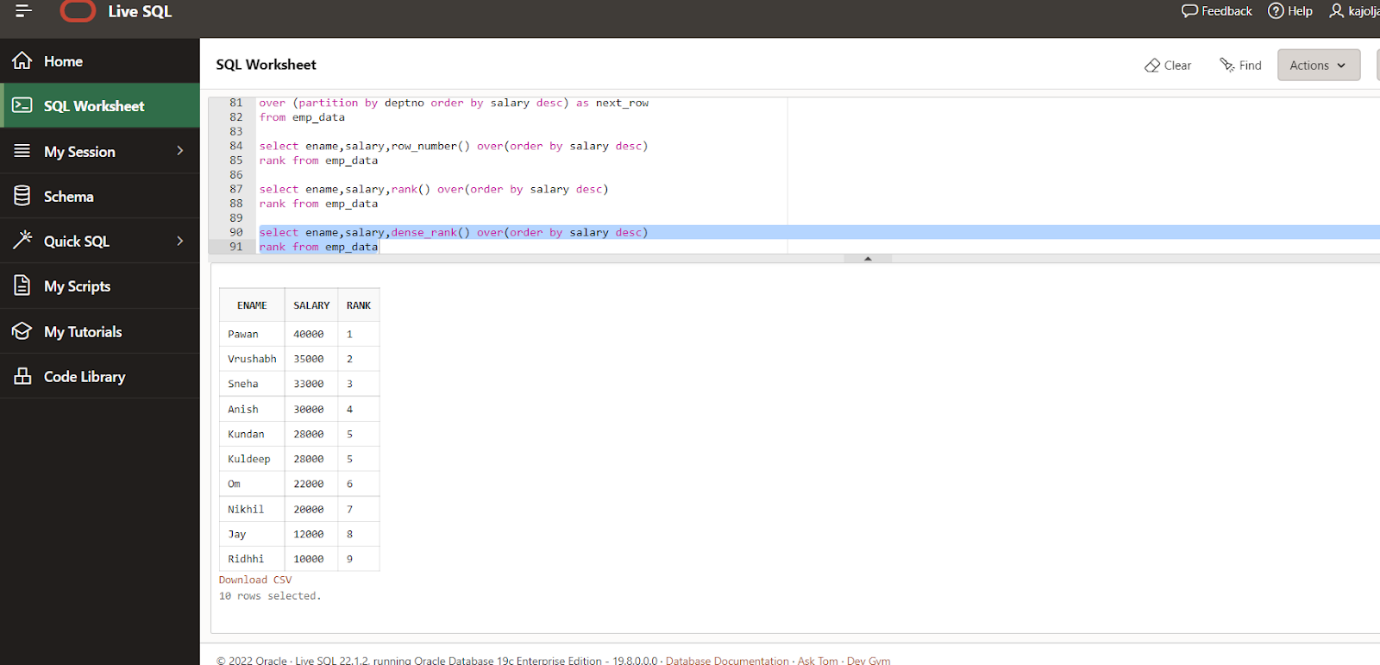
rank from emp\_data



Using dense\_rank

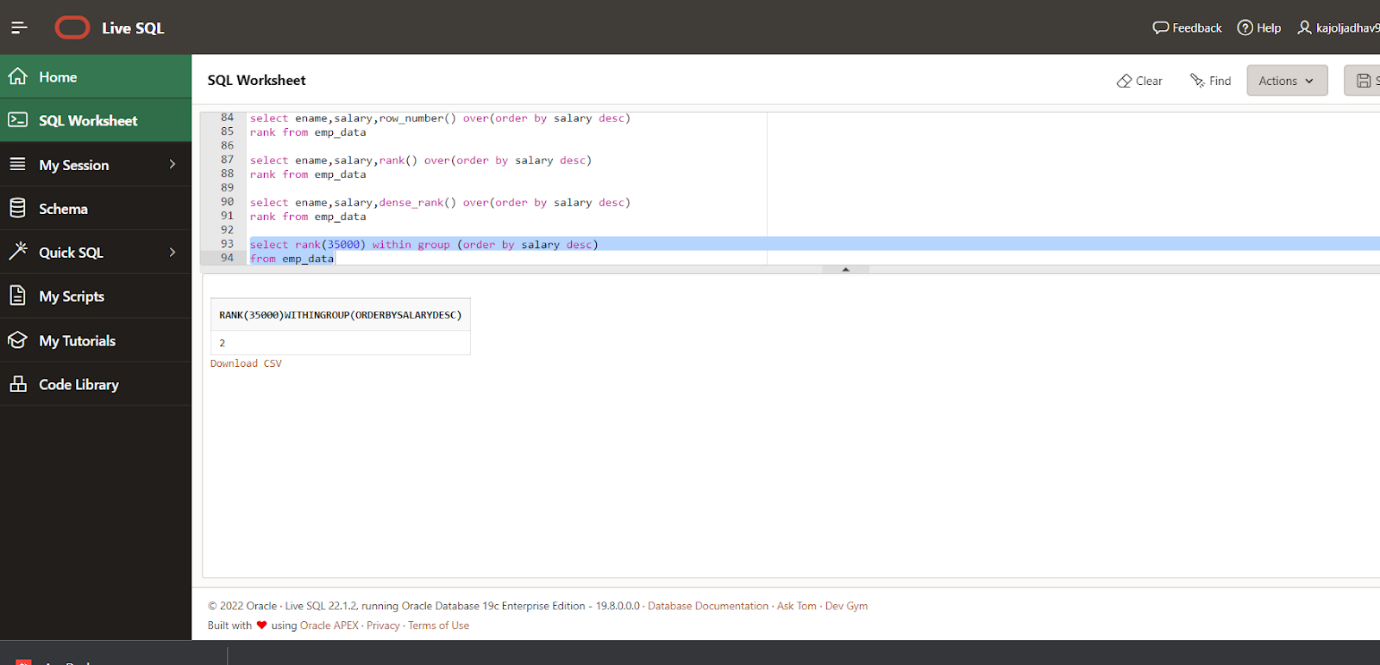
select ename,salary,dense\_rank() over(order by salary desc)

rank from emp\_data



select rank(35000) within group (order by salary desc)

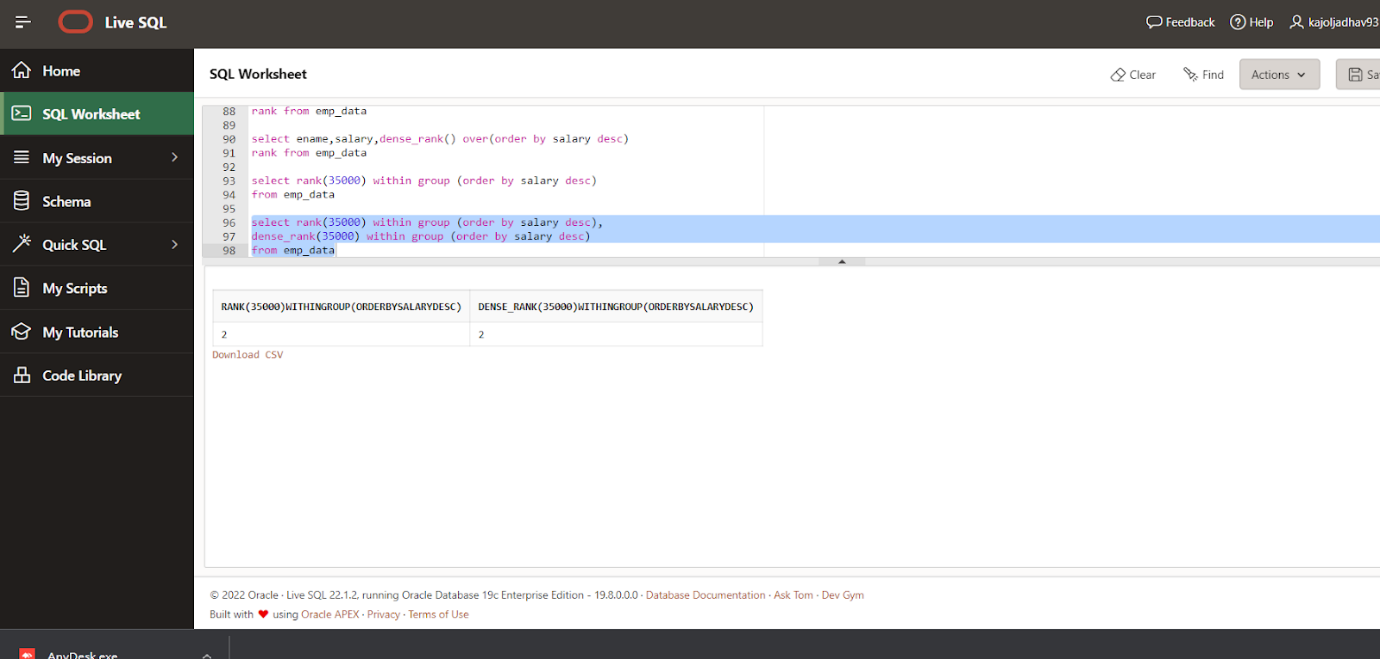
from emp\_data



select rank(35000) within group (order by salary desc),

dense\_rank(35000) within group (order by salary desc)

from emp\_data

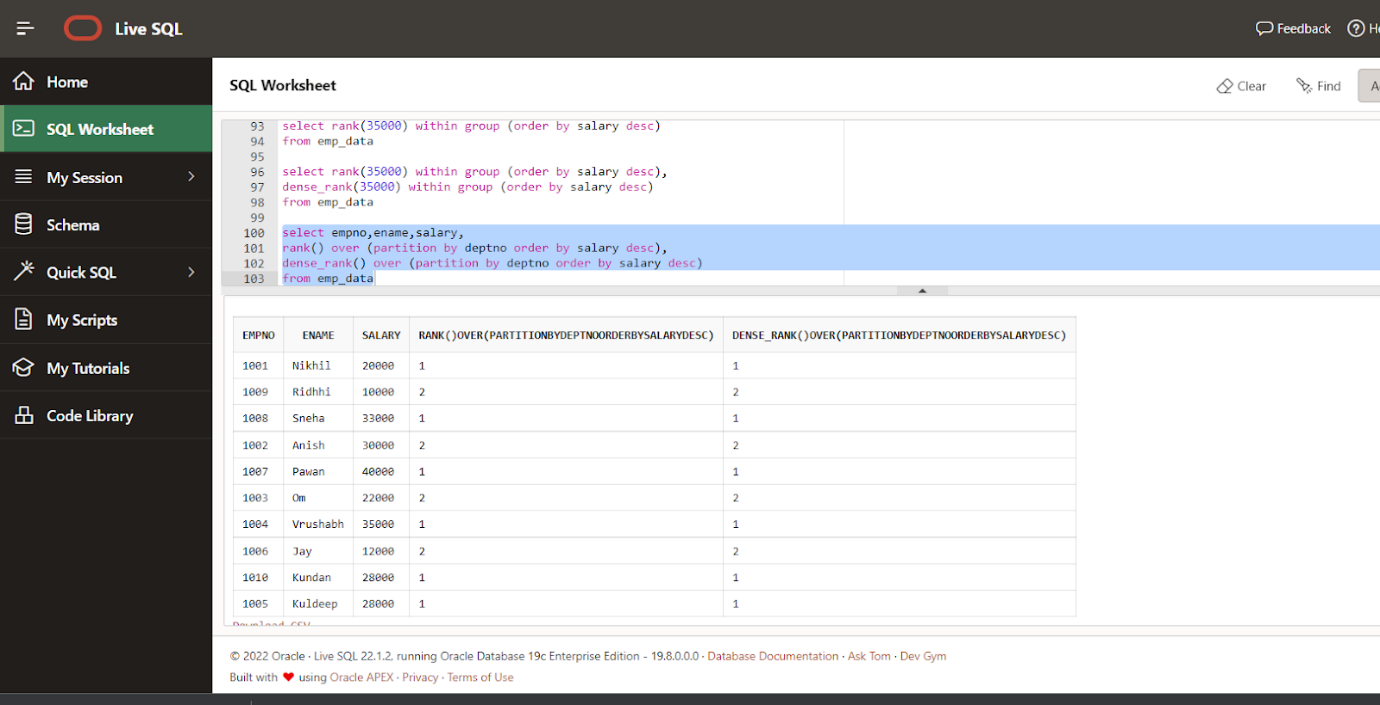


select empno,ename,salary,

rank() over (partition by deptno order by salary desc),

dense\_rank() over (partition by deptno order by salary desc)

from emp\_data

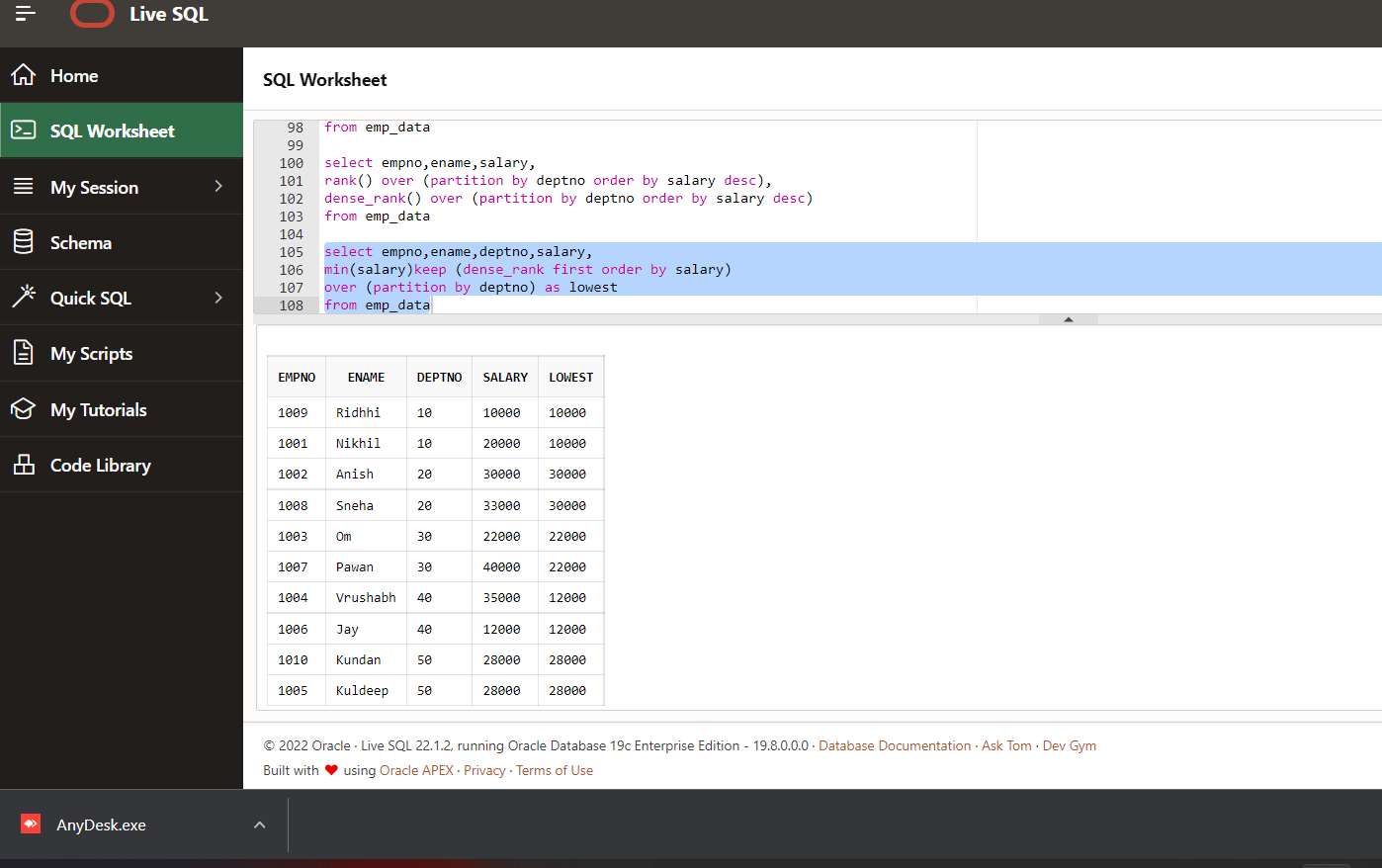


select empno,ename,deptno,salary,

min(salary)keep (dense\_rank first order by salary)

over (partition by deptno) as lowest

from emp\_data



select empno,ename,deptno,salary,

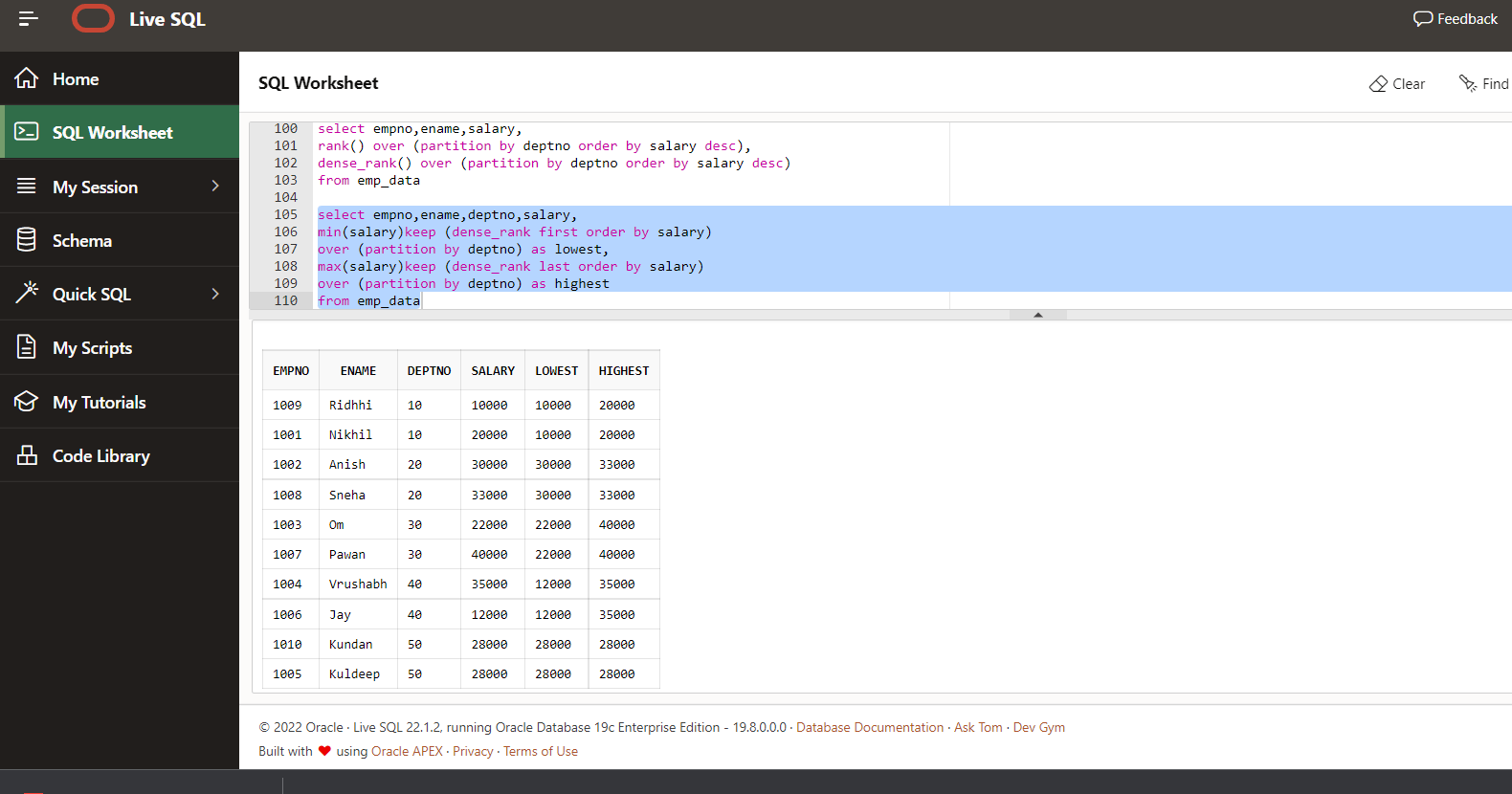
min(salary)keep (dense\_rank first order by salary)

over (partition by deptno) as lowest,

max(salary)keep (dense\_rank last order by salary)

over (partition by deptno) as highest

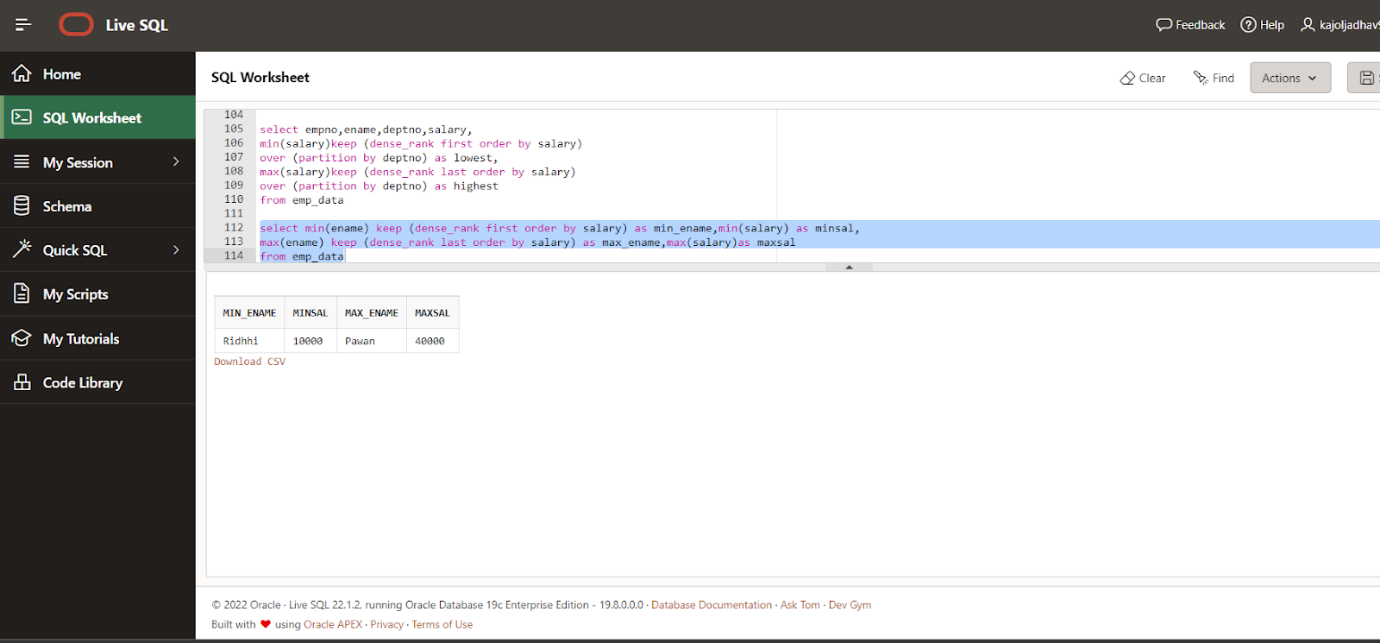
from emp\_data



select min(ename) keep (dense\_rank first order by salary) as min\_ename,min(salary) as minsal,

max(ename) keep (dense\_rank last order by salary) as max\_ename,max(salary)as maxsal

from emp\_data



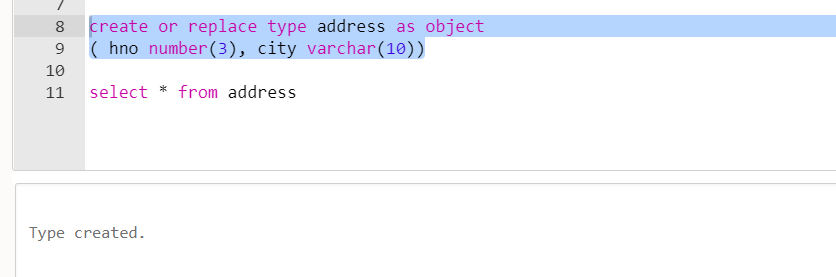
**Practical No.3**

**Aim: Implementation of ORDBMS concepts**

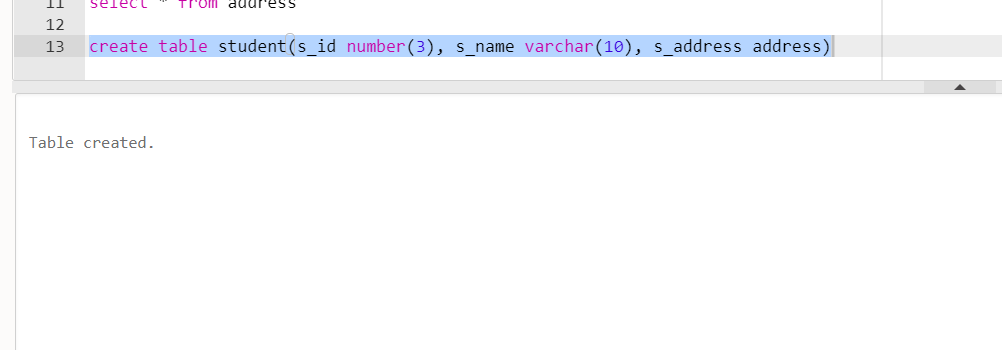
Code:

create or replace type address as object

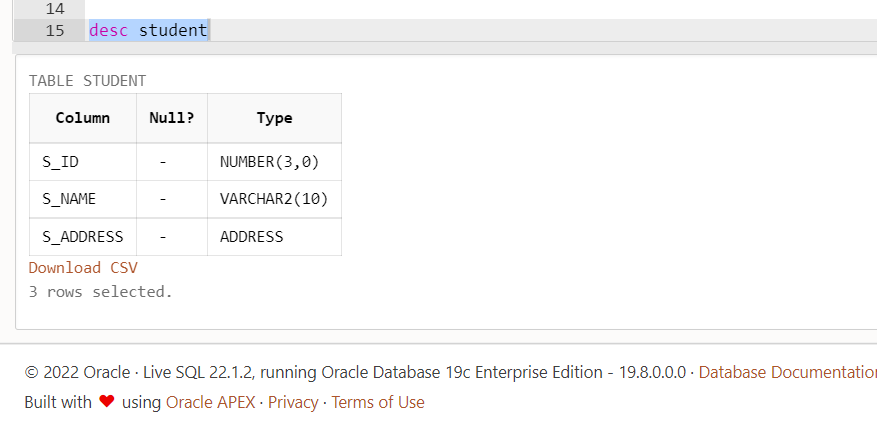
( hno number(3), city varchar(10))



create table student(s\_id number(3), s\_name varchar(10), s\_address address)



desc student



create table student\_type(s\_id number(3), s\_name varchar(10), s\_address address);

insert into student\_type values(1,'Anuj',address(101,'Thane'));

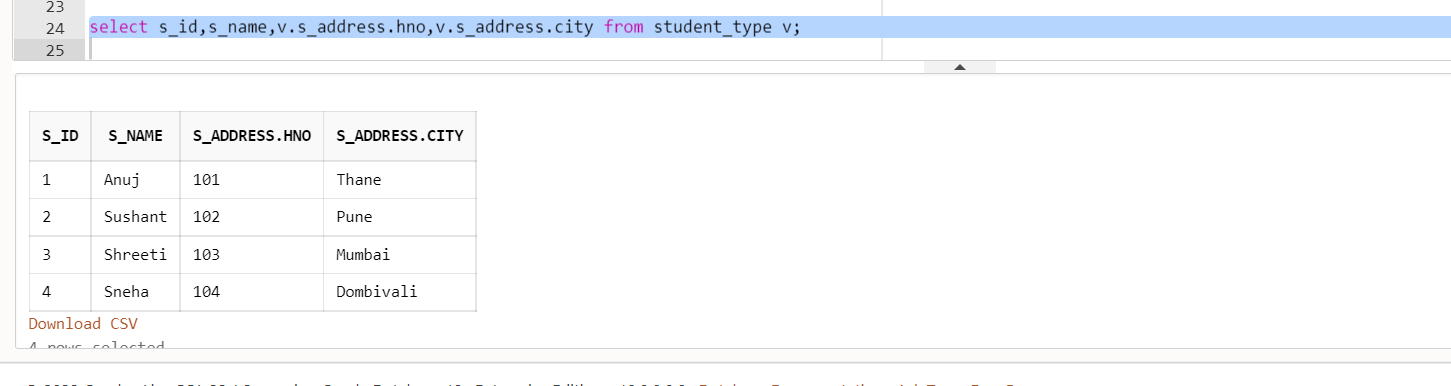
insert into student\_type values(2,'Sushant',address(102,'Pune'));

insert into student\_type values(3,'Shreeti',address(103,'Mumbai'));

insert into student\_type values(4,'Sneha',address(104,'Dombivali'));



select s\_id,s\_name,v.s\_address.hno,v.s\_address.city from student\_type v;

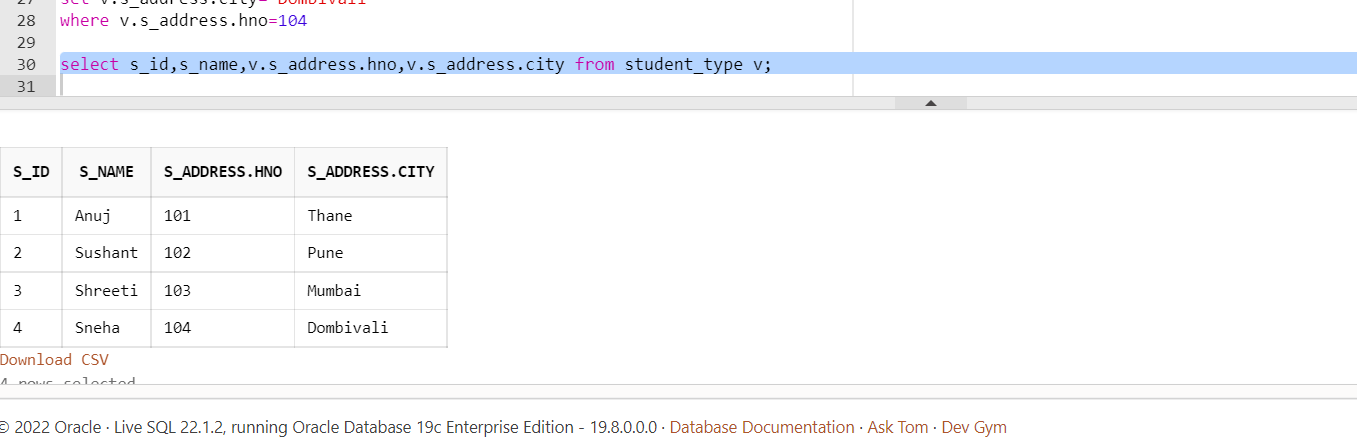


update student\_type v

set v.s\_address.city='Dombivali'

where v.s\_address.hno=104

select s\_id,s\_name,v.s\_address.hno,v.s\_address.city from student\_type v;



create type name as object

( first\_nm varchar(10),

last\_nm varchar(10))

create type full\_addr as object

( street varchar(10),

city varchar(10),

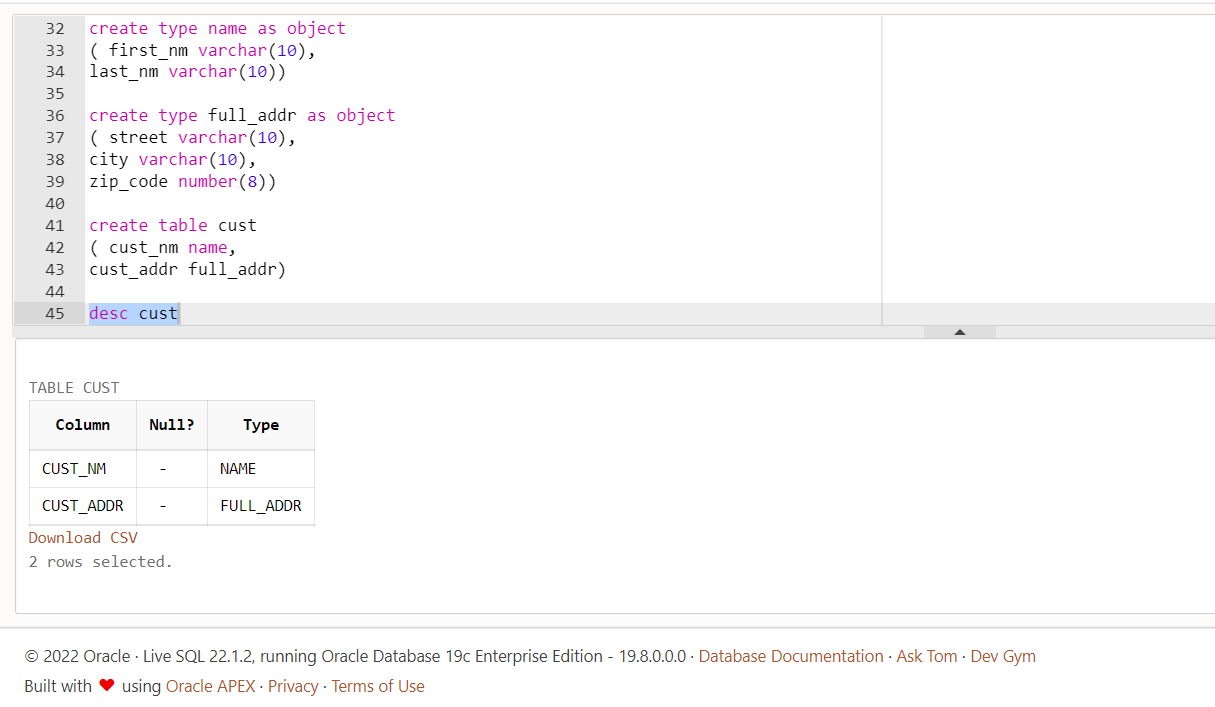
zip\_code number(8))

create table cust

( cust\_nm name,

cust\_addr full\_addr)

desc cust



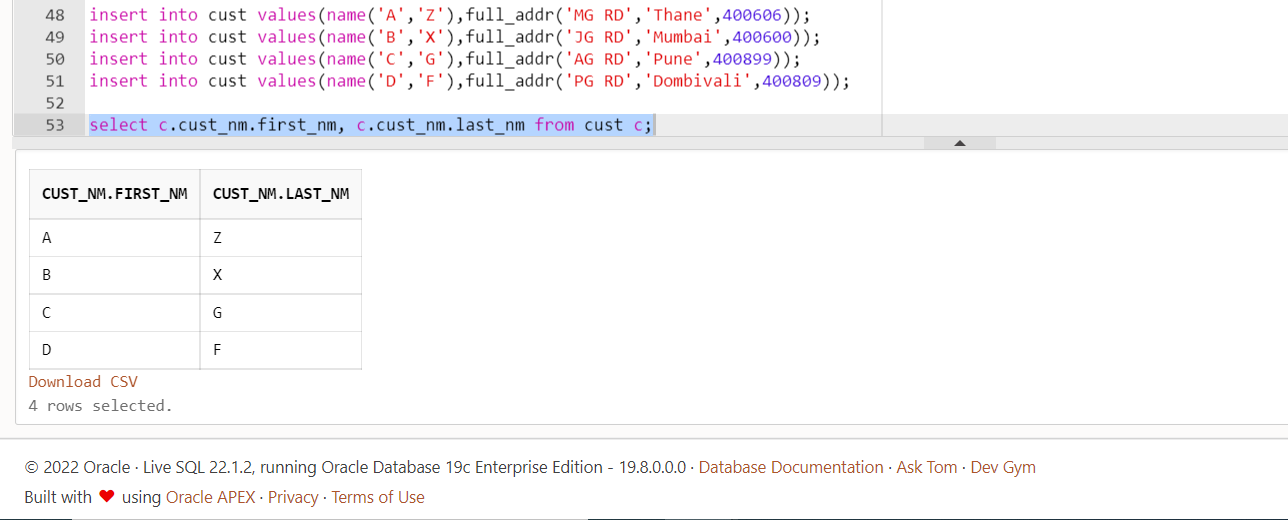
insert into cust values(name('A','Z'),full\_addr('MG RD','Thane',400606));

insert into cust values(name('B','X'),full\_addr('JG RD','Mumbai',400600));

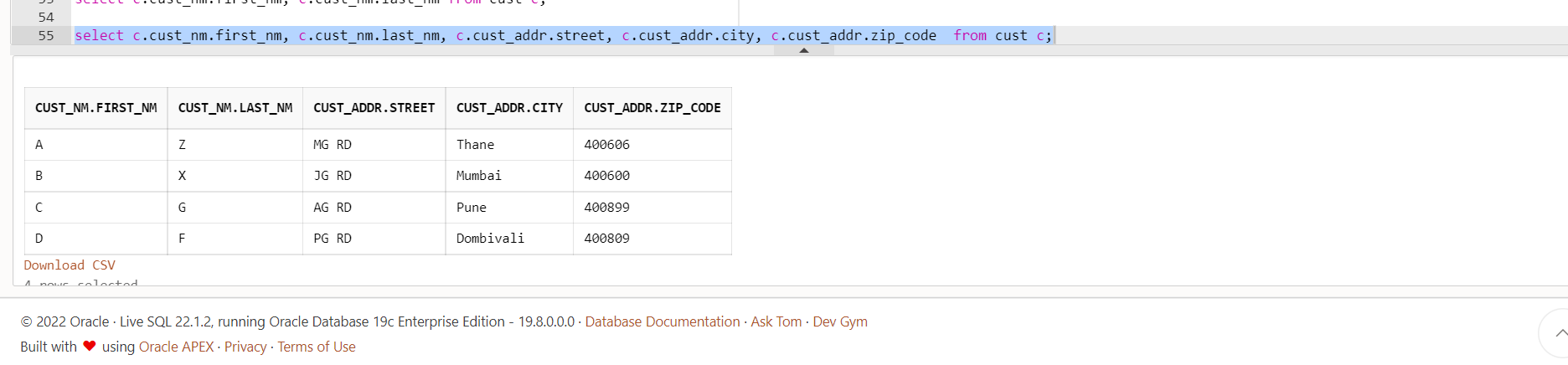
insert into cust values(name('C','G'),full\_addr('AG RD','Pune',400899));

insert into cust values(name('D','F'),full\_addr('PG RD','Dombivali',400809));

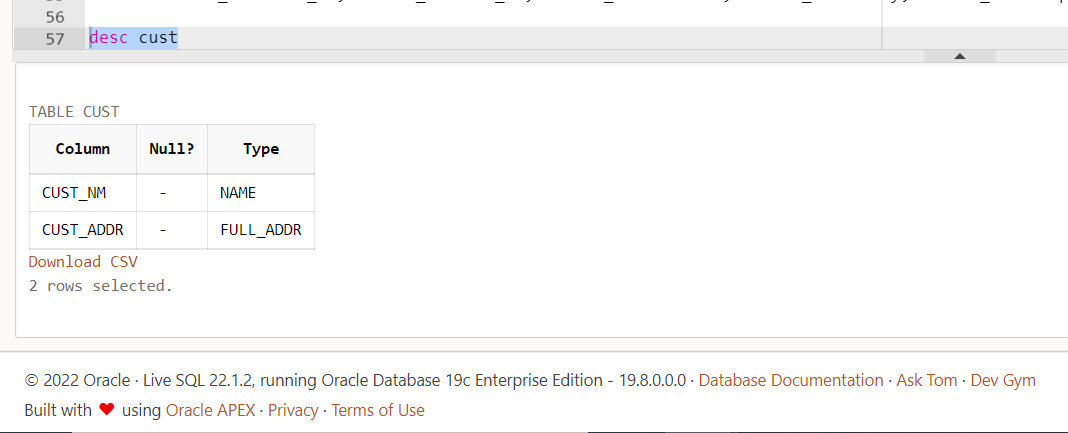
select c.cust\_nm.first\_nm, c.cust\_nm.last\_nm from cust c;



select c.cust\_nm.first\_nm, c.cust\_nm.last\_nm, c.cust\_addr.street, c.cust\_addr.city, c.cust\_addr.zip\_code  from cust c;



Desc cust



Q1. Create abstract data type type\_name with attributes fname,mname,lname.

Solution:

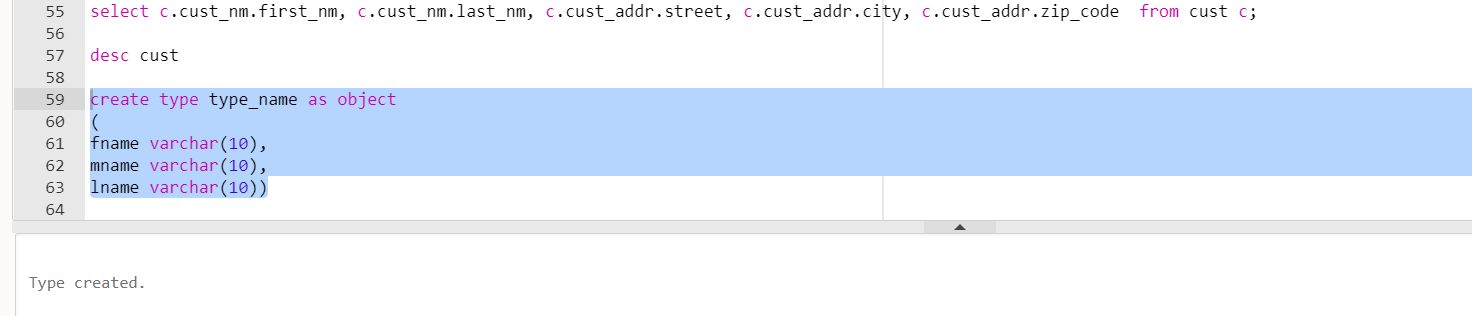
create type type\_name as object

(

fname varchar(10),

mname varchar(10),

lname varchar(10))



Q2. Create abstract data type type\_address with street,city,pincode.

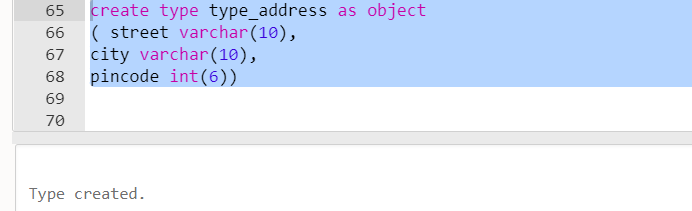
Solution:

create type type\_address as object

( street varchar(10),

city varchar(10),

pincode int(6))



Q3. Create table customer with type\_name and type\_address.

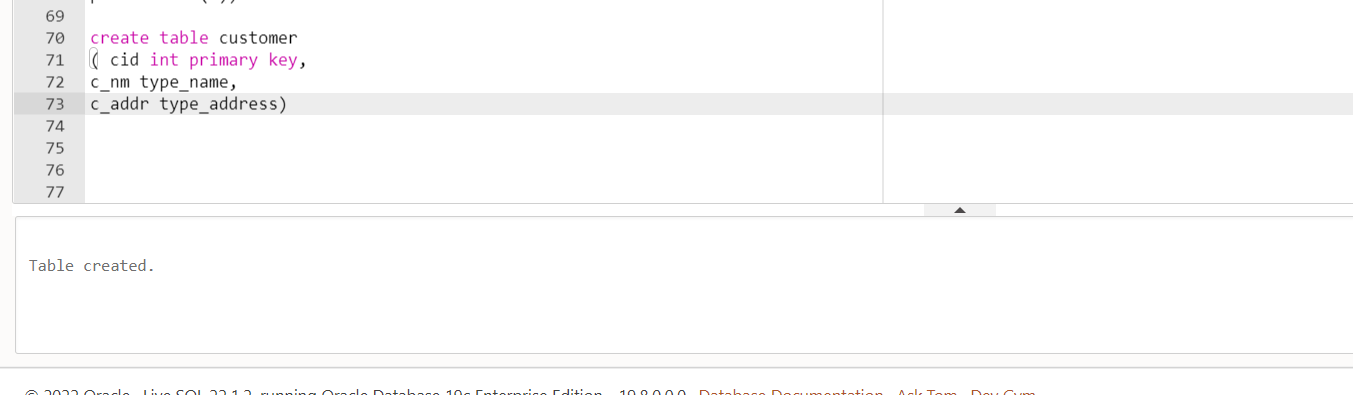
Solution:

create table customer

( cid int primary key,

c\_nm type\_name,

c\_addr type\_address)

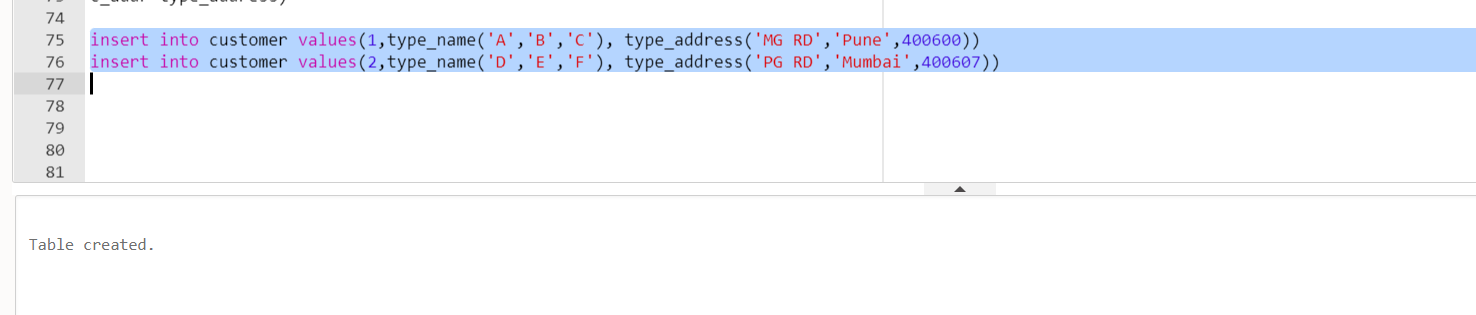


Q4.Insert Values into customer.

Solution:

insert into customer values(1,type\_name('A','B','C'), type\_address('MG RD','Pune',400600))

insert into customer values(2,type\_name('D','E','F'), type\_address('PG RD','Mumbai',400607))



Q5. Display the type\_name content.

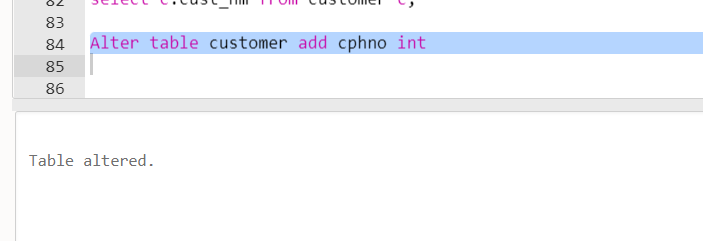
Solution:

select c.c\_nm from customer c;

Q6. Display cid,cphno from customer.

Solution:

Alter table customer add cphno int

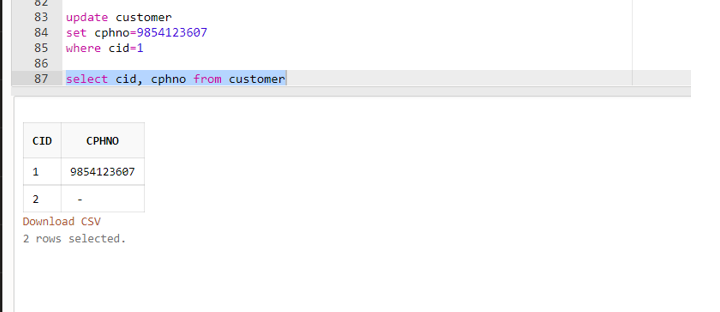


update customer

set cphno=9854123607

where cid=1

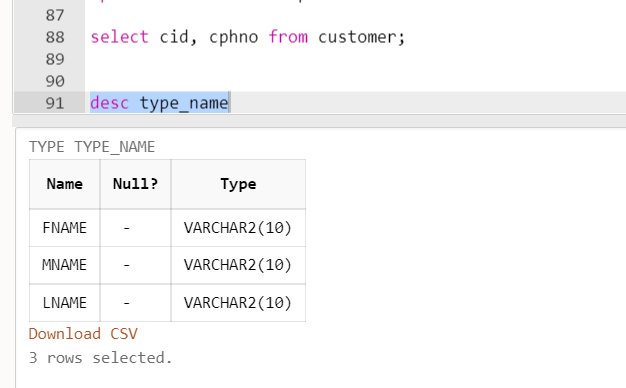
select cid, cphno from customer



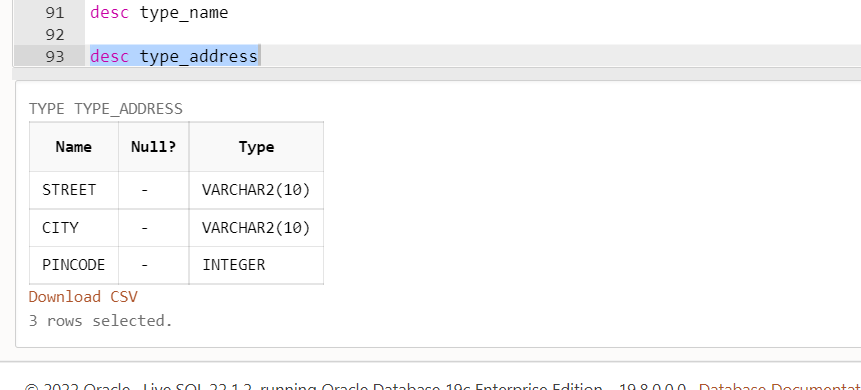
Q7. Describe type\_name and type\_address.

Solution:

desc type\_name



desc type\_address



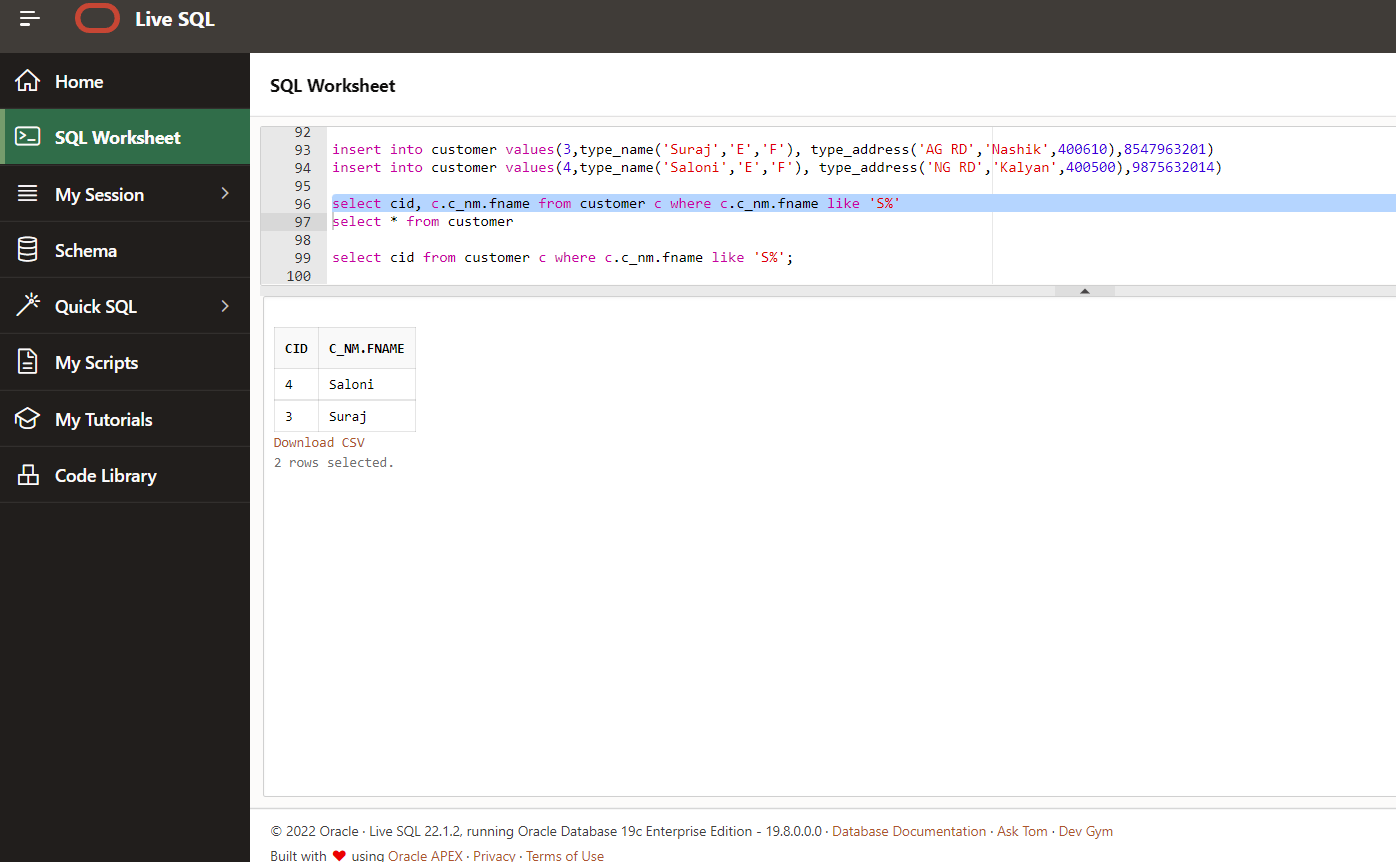
Q8. Display the fname from customer starts with s.

Solution:

insert into customer values(3,type\_name('Suraj','E','F'), type\_address('AG RD','Nashik',400610),8547963201)

insert into customer values(4,type\_name('Saloni','E','F'), type\_address('NG RD','Kalyan',400500),9875632014)

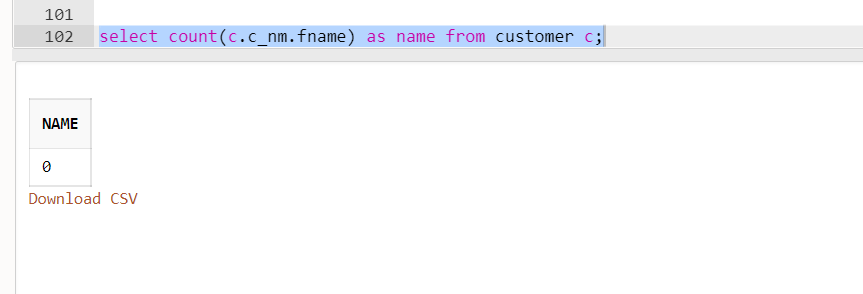
select cid, c.c\_nm.fname from customer c where c.c\_nm.fname like 'S%'



Q9. Display count of name from customer c.

Solution:

select count(c.c\_nm.fname) as name from customer c;



Inheritance

Q1. Create a type person\_type with attributes pid.pname & address

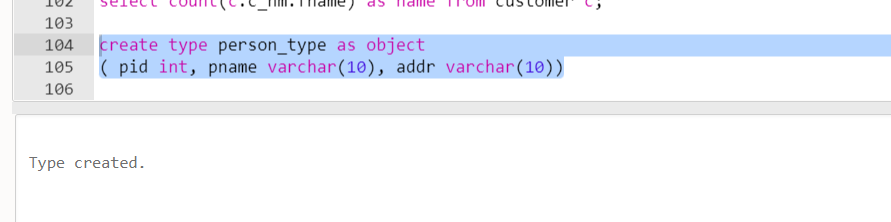
Solution:

create type person\_type as object

( pid int, pname varchar(10), addr varchar(10))

not final;

Output:



Q2. Create a type student\_type under person\_type with the attributes Deptname,major

Solution:

create type student\_type under person\_type

( dname varchar(10), major varchar(10))

not final;



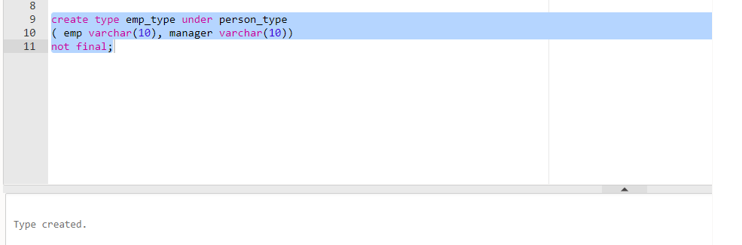
Q3. Create a type employe\_type under person\_type with the attributes emp.manager name

Solution:

create type emp\_type under person\_type

( emp varchar(10), manager varchar(10))

not final;



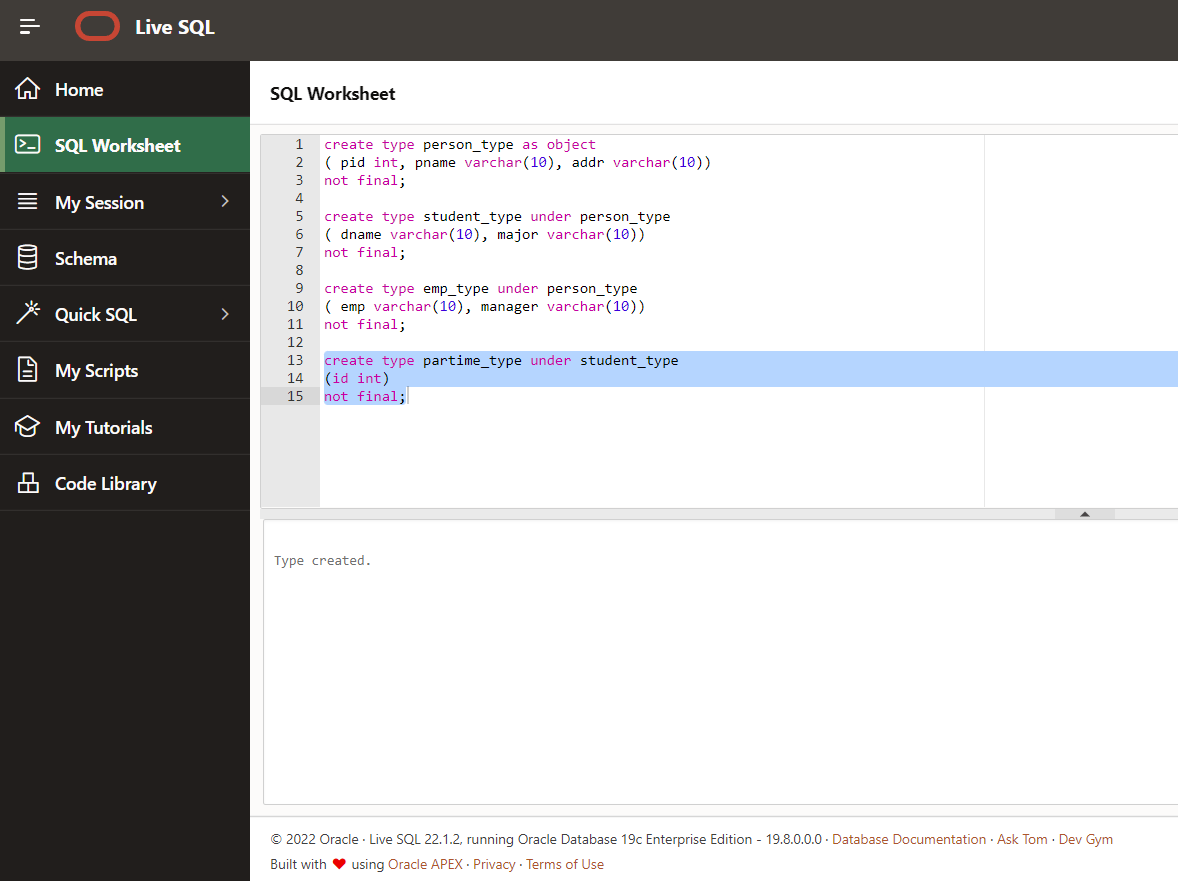
Q4. Create type partimestudent1\_type under student\_type

Solution:

create type partime\_type under student\_type

(id int)

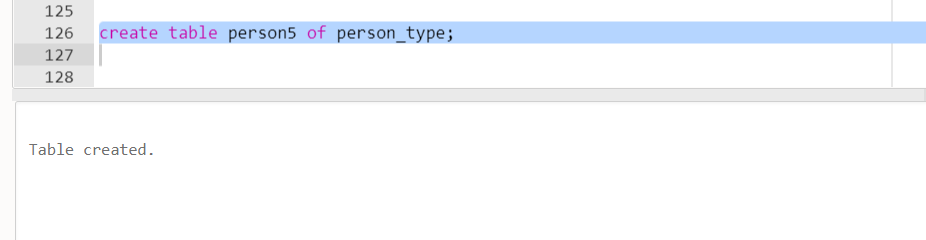
not final;



Q5. Create table person5 as an object table of person1\_type

Solution:

create table person5 of person\_type;

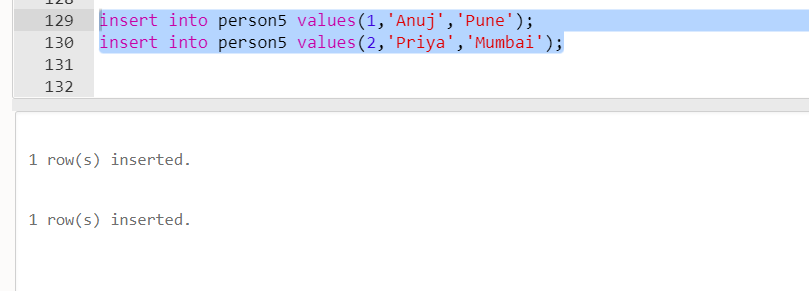


Q6. Insert data into person5

Solution:

insert into person5 values(1,Anuj,'Pune');

insert into person5 values(2,'Priya','Mumbai');



Q7. Display all details from person5

Solution:

select \* from person5;



Q8. Display values from person5

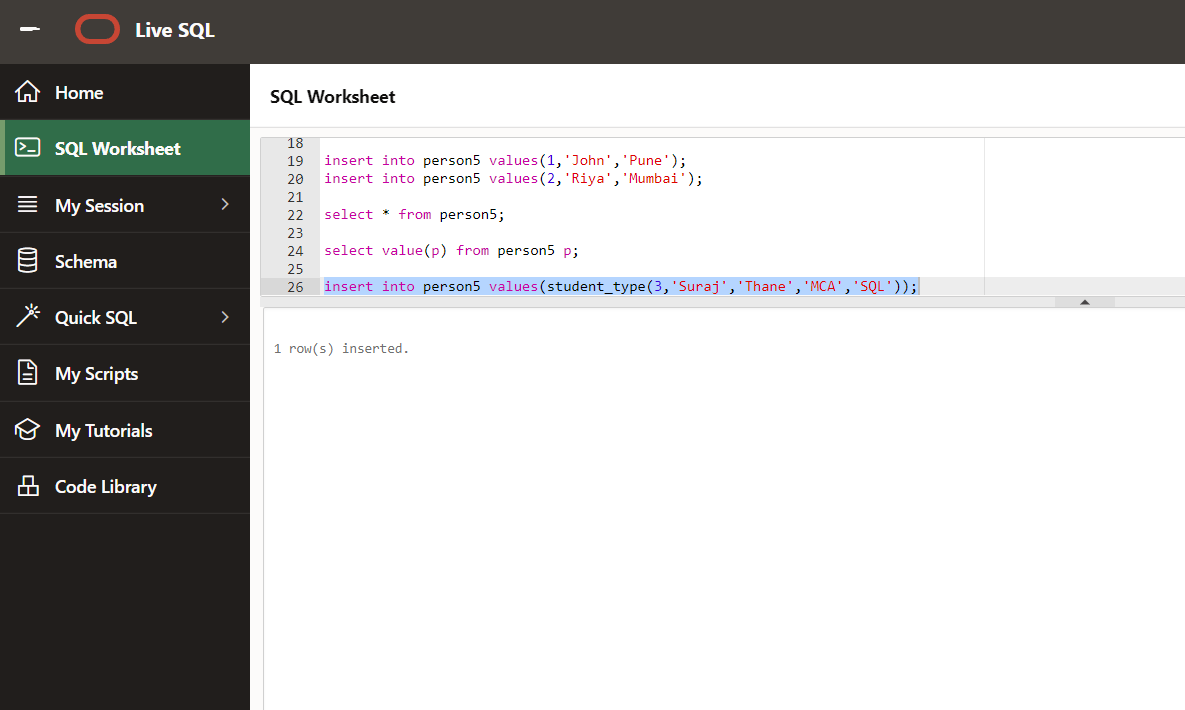
Solution:

select value(p) from person5 p;

Q9. Insert into student\_type

Solution:

insert into person5 values(student\_type(3,'Suraj','Thane','MCA','SQL'));



Refrences

create type ADDRESS as object

(street\_name varchar2(30),

house\_no number);

create table people(col1 varchar2(30),

col2 number,

col3 ref ADDRESS)

Deriving Sub-Tables from the Base Table usisng Refrences

create table persontype

(

persontypeid int primary key,

persontype varchar(20)

);

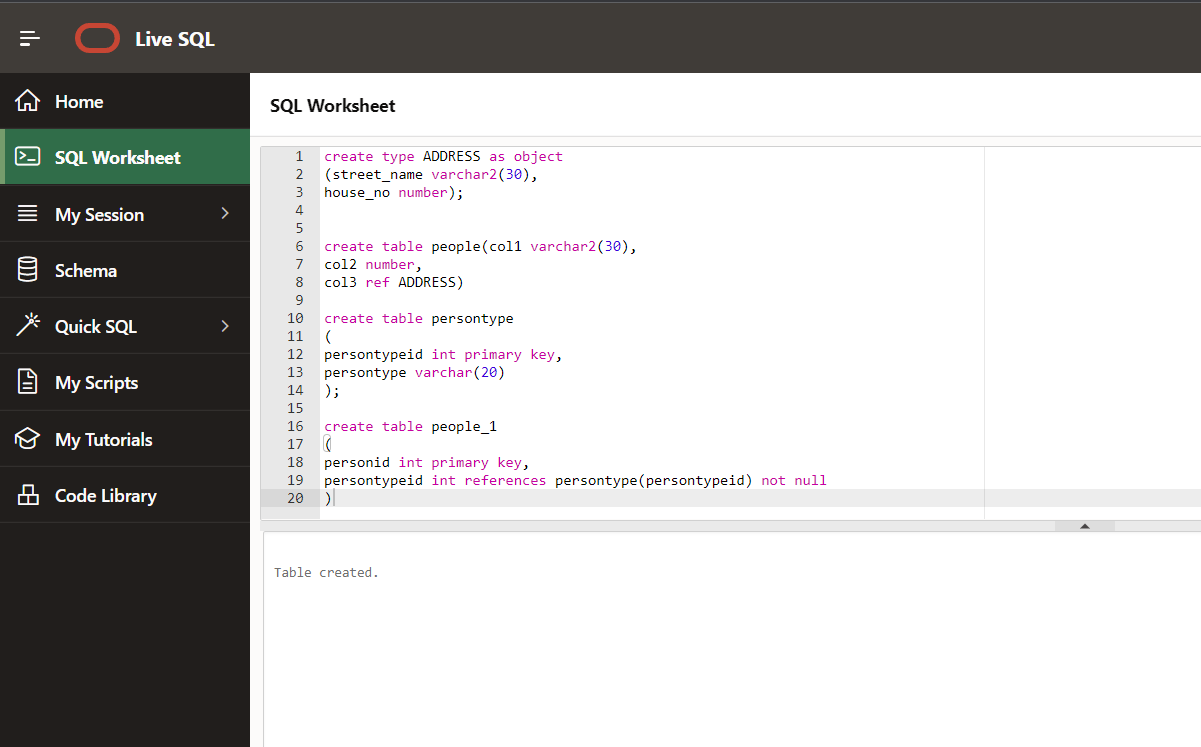
create table people\_1

(

personid int primary key,

persontypeid int references persontype(persontypeid) not null

)



PLSQL

create or replace type address\_1 as object

(house\_no varchar(10),

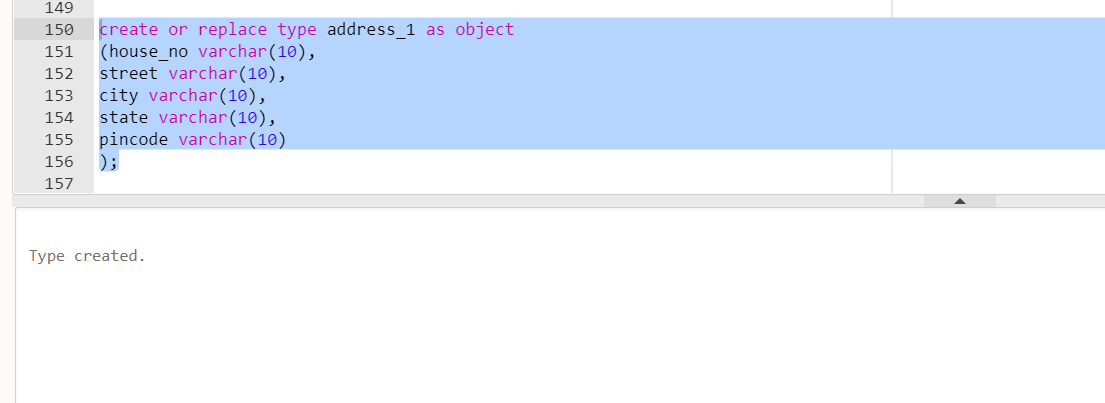
street varchar(10),

city varchar(10),

state varchar(10),

pincode varchar(10)

);



DECLARE

  residence address\_1;

BEGIN

  residence := address\_1('103A', 'M.G ROAD', 'Thane', 'Mumbai', '201301');

  dbms\_output.put\_line('house No: '|| residence.house\_no);

  dbms\_output.put\_line('street: '|| residence.house\_no);

  dbms\_output.put\_line('city: '|| residence.house\_no);

  dbms\_output.put\_line('state: '|| residence.house\_no);

  dbms\_output.put\_line('pincode: '|| residence.house\_no);

END;

Output:



**Calculator**

create type myinput2 as object(num1 int,num2 int);

create type mycalculator as object(

mynum myinput2,

member function plus return number,

member function sub return number,

member function mul return number,

member function div return number

);

create type body mycalculator as

member function plus return number as result number;

BEGIN

result:=mynum.num1+mynum.num2;

return result;

END plus;

member function sub return number as result number;

BEGIN

result:=mynum.num1-mynum.num2;

return result;

END sub;

member function mul return number as result number;

BEGIN

result:=mynum.num1\*mynum.num2;

return result;

END mul;

member function div return number as result number;

BEGIN

result:=mynum.num1/mynum.num2;

return result;

END div;

END;

DECLARE

obj1 myinput2 := myinput2(40,20);

obj2 mycalculator := mycalculator(obj1);

BEGIN

dbms\_output.put\_line('Addition: '||obj2.Plus);

dbms\_output.put\_line('Subraction: '||obj2.Sub);

dbms\_output.put\_line('Multiplication: '||obj2.Mul);

dbms\_output.put\_line('Fivision: '||obj2.Div);

END; (num1 int,

num2 int);

create type MyCal1 as object

(

mynum MyInput,

Member Function Plus return number,

Member Function Sub return number,

Member Function Mul return number,

Member Function Div return number

);

create type body MyCal as

Member function Plus return number as

Result number;

Begin

result := mynum.num1+mynum.num2;

Return result;

End Plus;

Member function Sub return number as

Result number;

Begin

result := mynum.num1-mynum.num2;

Return result;

End Sub;

Member function Mul return number as

Result number;

Begin

result := mynum.num1\*mynum.num2;

Return result;

End Mul;

Member function Div return number as

Result number;

Begin

result := mynum.num1/mynum.num2;

Return result;

End Div;

End;

Declare

obj1 MyInput := MyInput(50,100);

obj2 MyCal := MyCal(obj1);

Begin

DBMS\_Output.Put\_Line('Addition : ' || obj2.Plus);

DBMS\_Output.Put\_Line('Subtraction : ' || obj2.Sub);

DBMS\_Output.Put\_Line('Multiplication : ' || obj2.Mul);

DBMS\_Output.Put\_Line('Division : ' || obj2.Div);

End;

