

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Programme | : | **M.Tech. (Integrated) Software Engineering** | Semester | : | **Fall 20-21** |
| Course | : | **Advanced Database Management Systems** | Code | : | **SWE2014** |
| Faculty | : | **Dr.Bhuvaneswari A** | Slot | : | **L27 + L28** |
| Date | : | **04-08-2020** | Marks | : | **10 Points** |

**NAME:** K. NAVEEN KUMAR

**REG NO:** 17MIS1056

Ex No. 4 Advanced SQL- Ordinary Views and Materialized Views

Create table for the following schema:

SALESMAN (Salesman\_id, Name, City, Commission)

CUSTOMER (Customer\_id, Cust\_Name, City, Grade, Salesman\_id)

ORDERS (Ord\_No, Purchase\_Amt, Ord\_Date, Customer\_id, Salesman\_id)

Sample table: Salesman

salesman\_id | name | city | commission

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

5001 | James Hoog | New York | 0.15

5002 | Nail Knite | Paris | 0.13

5005 | Pit Alex | London | 0.11

5006 | Mc Lyon | Paris | 0.14

5007 | Paul Adam | Rome | 0.13

5003 | Lauson Hen | San Jose | 0.12

Sample table: Customer

customer\_id | cust\_name | city | grade | salesman\_id

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

3002 | Nick Rimando | New York | 100 | 5001

3007 | Brad Davis | New York | 200 | 5001

3005 | Graham Zusi | California | 200 | 5002

3008 | Julian Green | London | 300 | 5002

3004 | Fabian Johnson | Paris | 300 | 5006

3009 | Geoff Cameron | Berlin | 100 | 5003

3003 | Jozy Altidor | Moscow | 200 | 5007

3001 | Brad Guzan | London | | 5005

Sample table: orders

ord\_no purchase\_amt ord\_date customer\_id salesman\_id

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

70001 150.5 2012-10-05 3005 5002

70009 270.65 2012-09-10 3001 5005

70002 65.26 2012-10-05 3002 5001

70004 110.5 2012-08-17 3009 5003

70007 948.5 2012-09-10 3005 5002

70005 2400.6 2012-07-27 3007 5001

70008 5760 2012-09-10 3002 5001

70010 1983.43 2012-10-10 3004 5006

70003 2480.4 2012-10-10 3009 5003

70012 250.45 2012-06-27 3008 5002

70011 75.29 2012-08-17 3003 5007

70013 3045.6 2012-04-25 3002 5001

**TABLE CREATIONS:**

SALESMAN TABLE:

create table salesman(

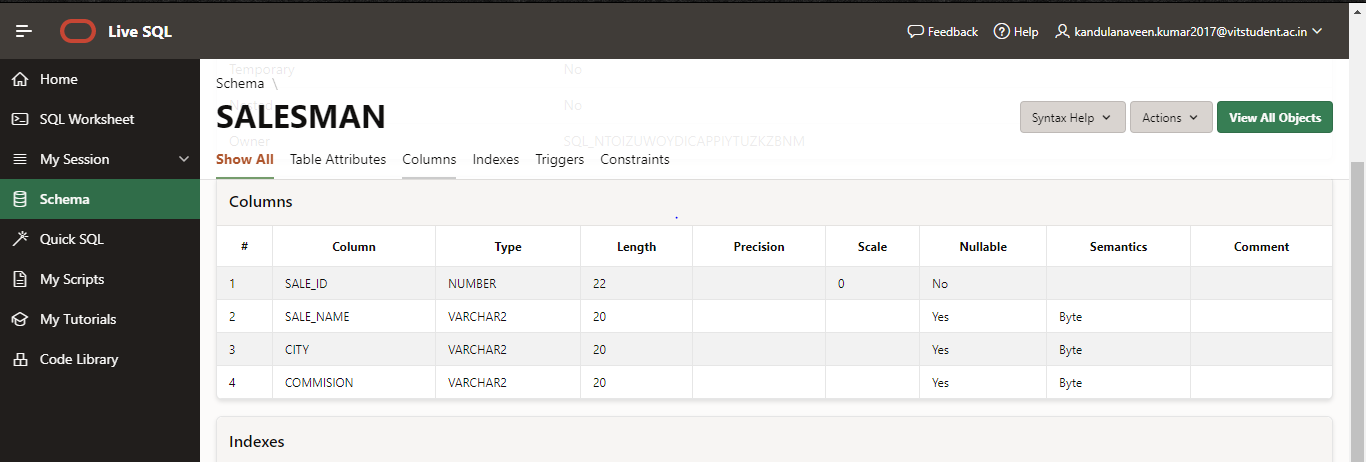
sale\_id integer primary key,

sale\_name varchar(20),

City varchar(20),

Commision varchar(20)

);



CUSTOMER TABLE:

create table customer(

cust\_id integer primary key,

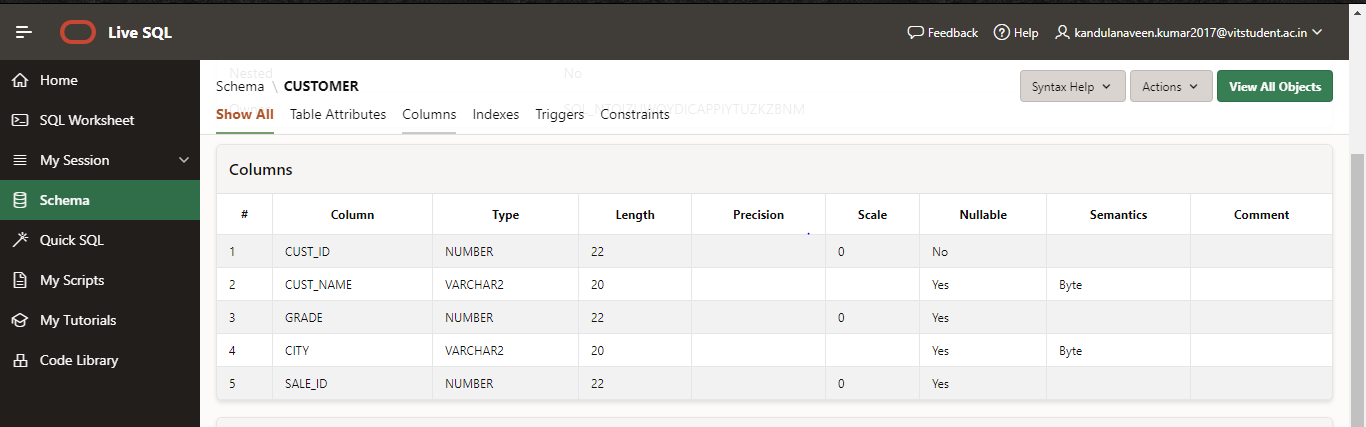
cust\_name varchar(20),

grade integer,

City varchar(20),

sale\_id integer,

foreign key (sale\_id) references salesman(sale\_id));



ORDERS TABLE:

create table orders(

order\_no integer primary key,

amt number(20),

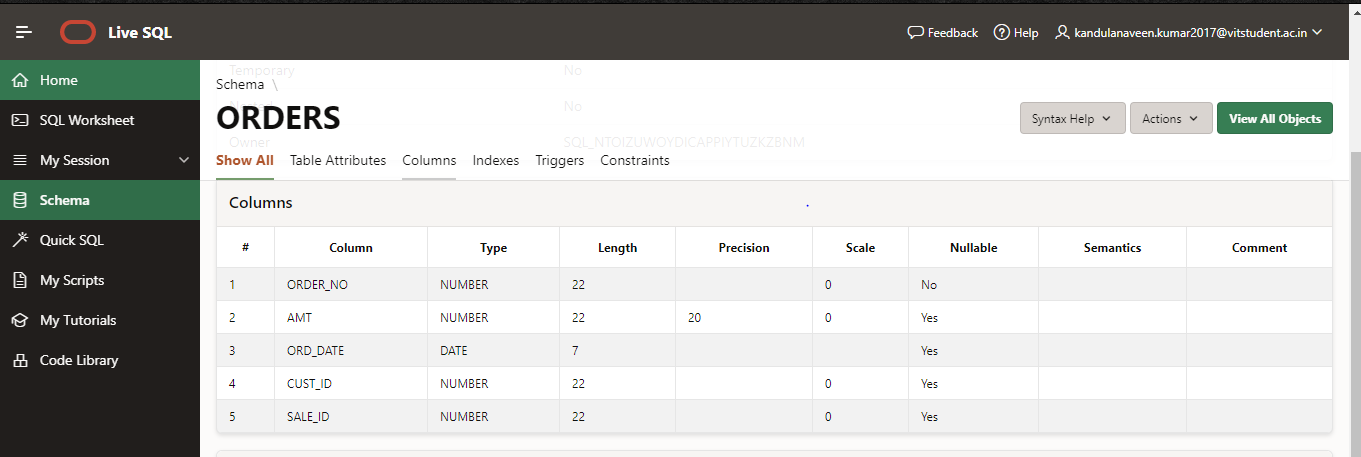
ord\_date date,

cust\_id integer,

sale\_id integer,

foreign key (sale\_id) references salesman(sale\_id),

foreign key (cust\_id) references customer(cust\_id));



**INSERTING THE VALUES:**

SALESMAN TABLE:

insert into salesman values(5001,'James Hoog','New York',0.15);

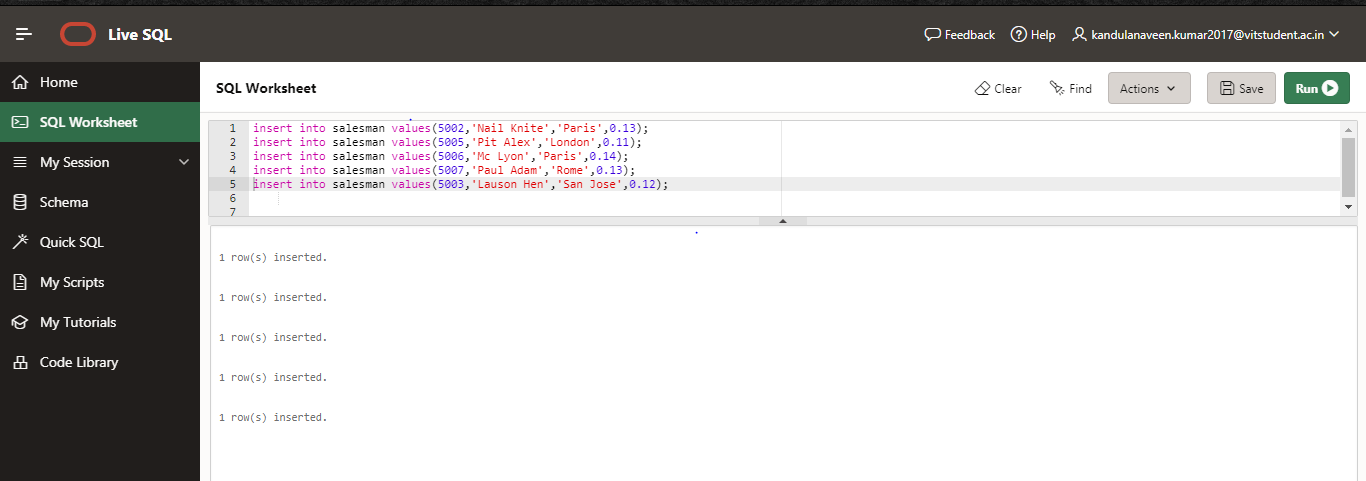
insert into salesman values(5002,'Nail Knite','Paris',0.13);

insert into salesman values(5005,'Pit Alex','London',0.11);

insert into salesman values(5006,'Mc Lyon','Paris',0.14);

insert into salesman values(5007,'Paul Adam','Rome',0.13);

insert into salesman values(5003,'Lauson Hen','San Jose',0.12);



CUSTOMER TABLE:

insert into customer values(3002,'Nick Rimando',100,'New York',5001);

insert into customer values(3007,'Brad Davis',200,'New York',5001);

insert into customer values(3005,'Graham Suzi',200,'California',5002);

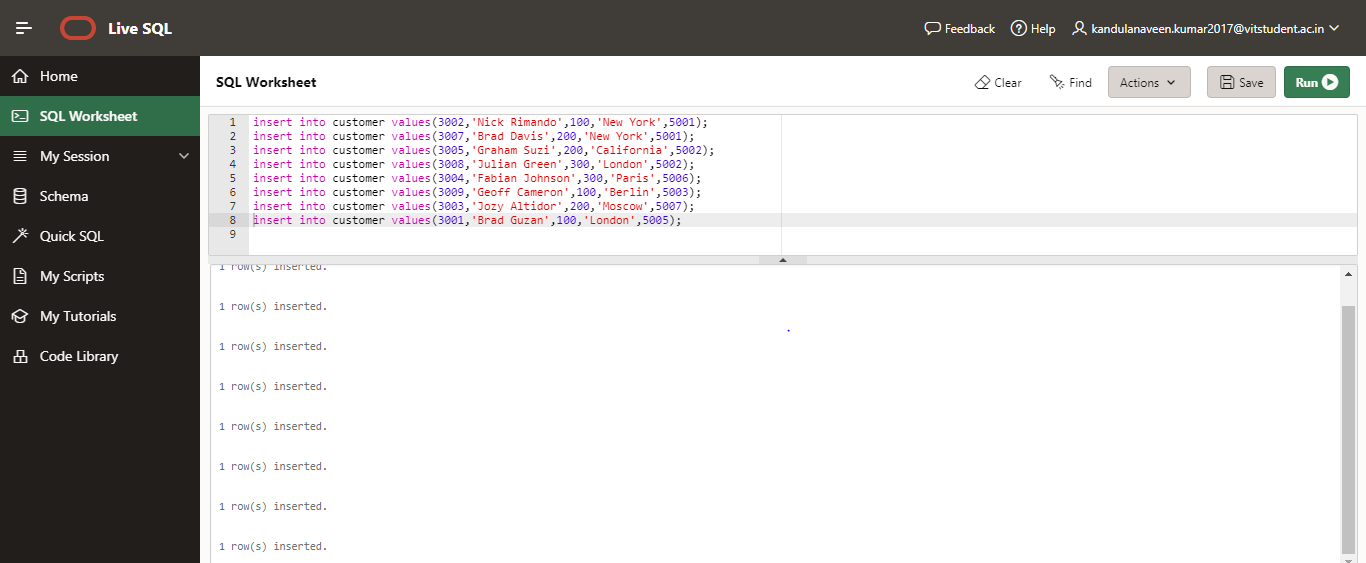
insert into customer values(3008,'Julian Green',300,'London',5002);

insert into customer values(3004,'Fabian Johnson',300,'Paris',5006);

insert into customer values(3009,'Geoff Cameron',100,'Berlin',5003);

insert into customer values(3003,'Jozy Altidor',200,'Moscow',5007);

insert into customer values(3001,'Brad Guzan',100,'London',5005);



ORDERS TABLE:

insert into orders values(70001,150,date'10-03-12',3005,5002);

insert into orders values(70009,270.65,date'10-03-12',3001,5005);

insert into orders values(70002,65.26,date'10-03-12',3002,5001);

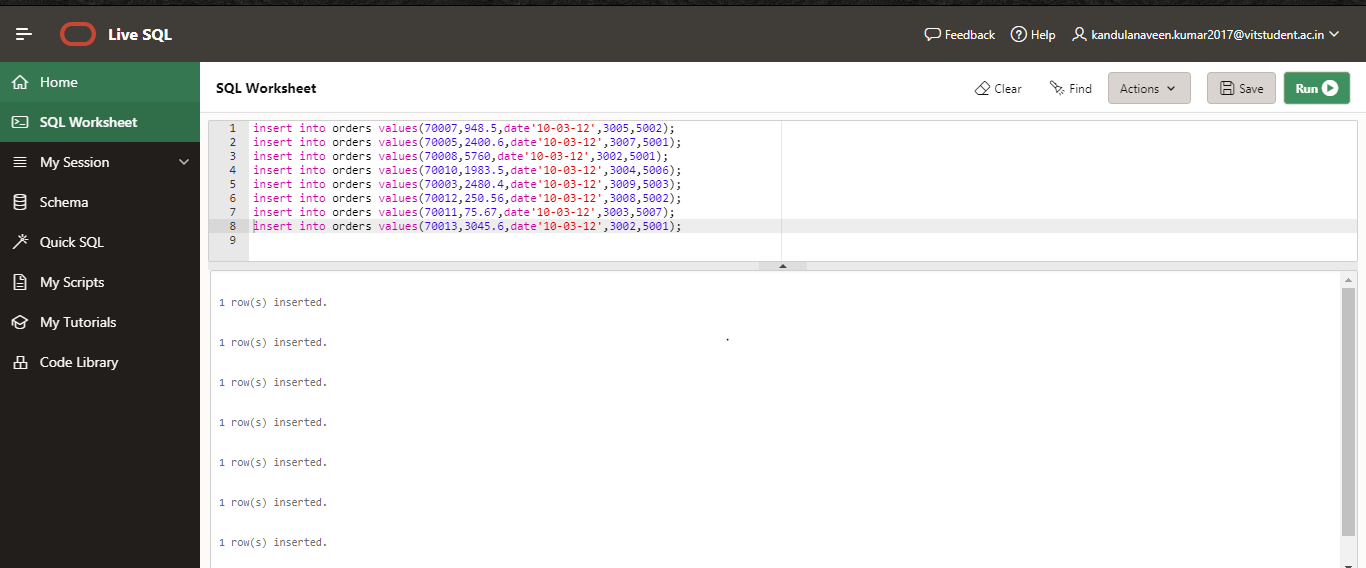
insert into orders values(70004,110.5,date'10-03-12',3009,5003);

insert into orders values(70001,150,date'10-03-12',3005,5002);

insert into orders values(70009,270.65,date'10-03-12',3001,5005);

insert into orders values(70002,65.26,date'10-03-12',3002,5001);

insert into orders values(70004,110.5,date'10-03-12',3009,5003);



Write SQL queries to

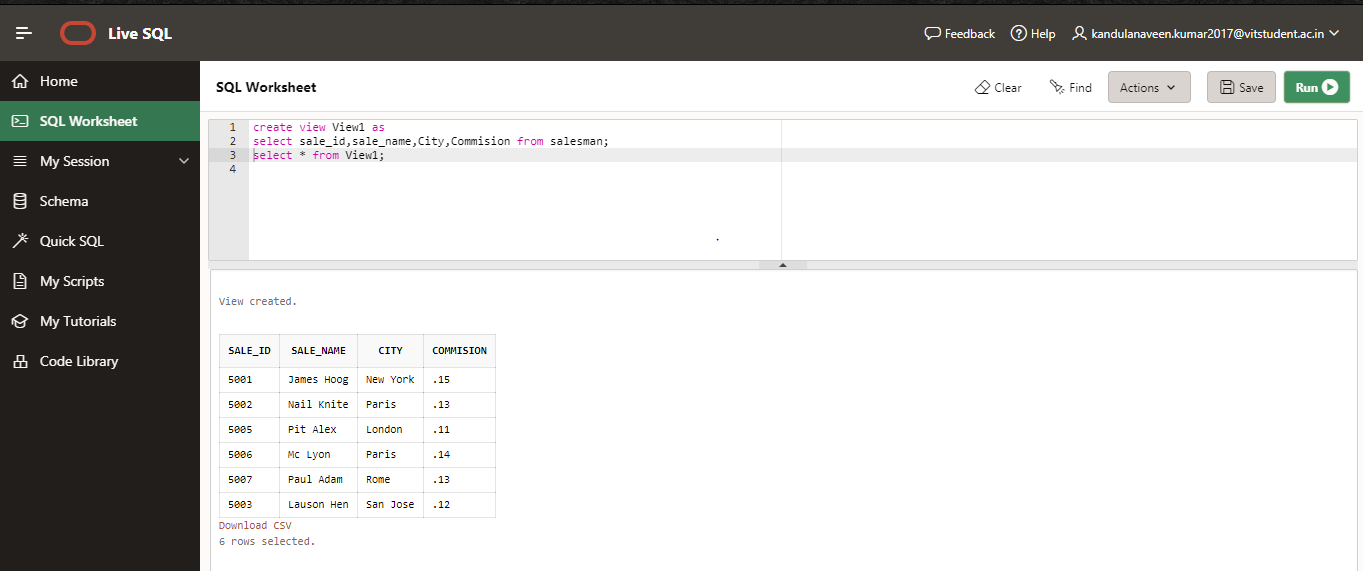
1. Create a simple view VIEW1 to display the salesman\_id, name , city , commission.
   1. Update the VIEW1 name of salesman with id 5001

ANS:

create view View1 as

select sale\_id,sale\_name,City,Commision from salesman;

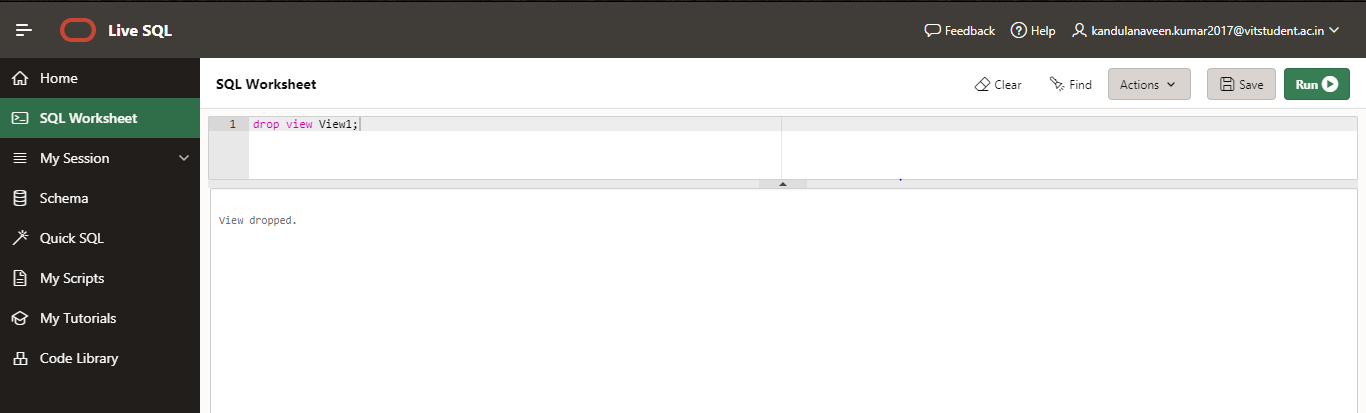
select \* from View1;



b.Drop the VIEW1

ANS:

drop view View1;



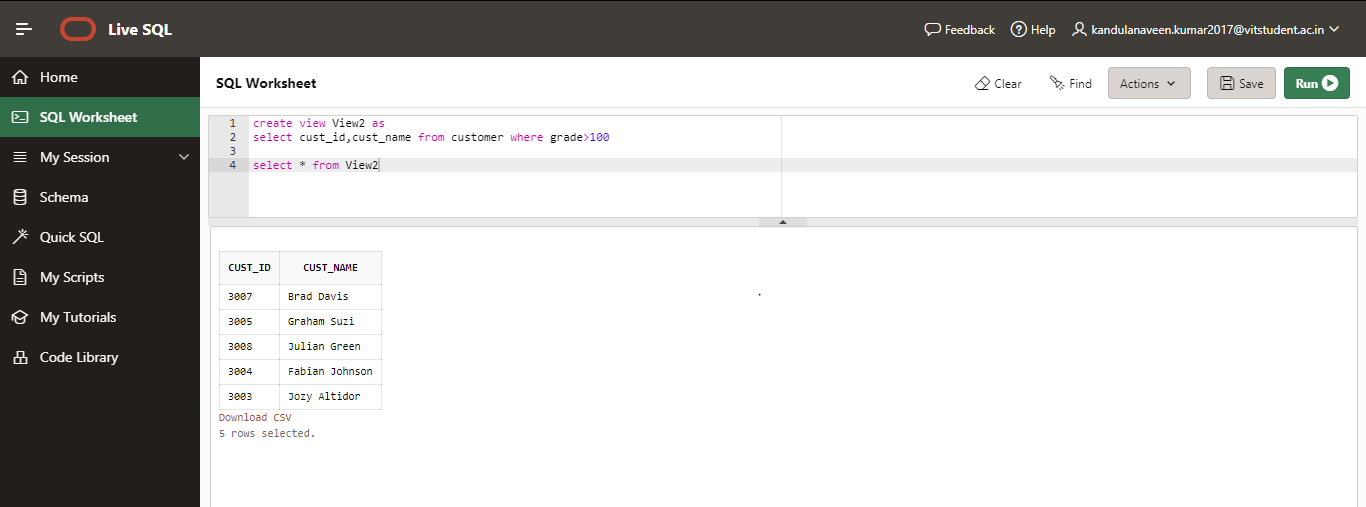
1. Create a simple view display the customers with grade more than 100 with Check option on Not Null value on all records.

ANS:

create view View2 as

select cust\_id,cust\_name from customer where grade>100

select \* from View2



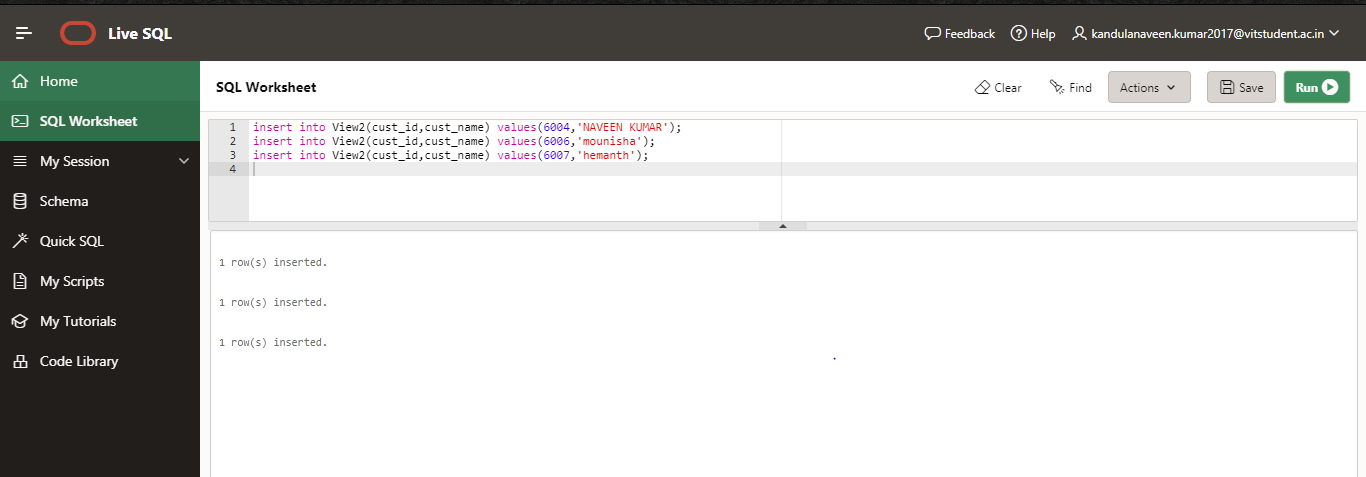
* 1. Add new rows into the view table

ANS:

insert into View2(cust\_id,cust\_name) values(6004,'NAVEEN KUMAR');

insert into View2(cust\_id,cust\_name) values(6006,'mounisha');

insert into View2(cust\_id,cust\_name) values(6007,'hemanth');



* 1. Delete one row from the view table

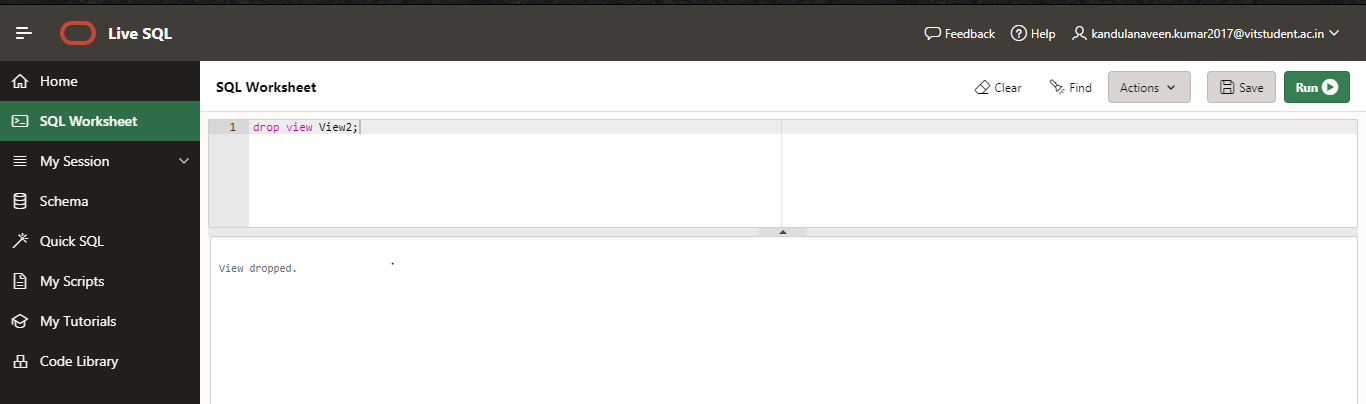
ANS:

delete from View2 where cust\_id in (6004,6006,6007);

* 1. Drop the VIEW

ANS:

drop view View2;



1. Create a simple view VIEW 2 that Find the name and numbers of all salesmen who had more than one customer.

ANS:

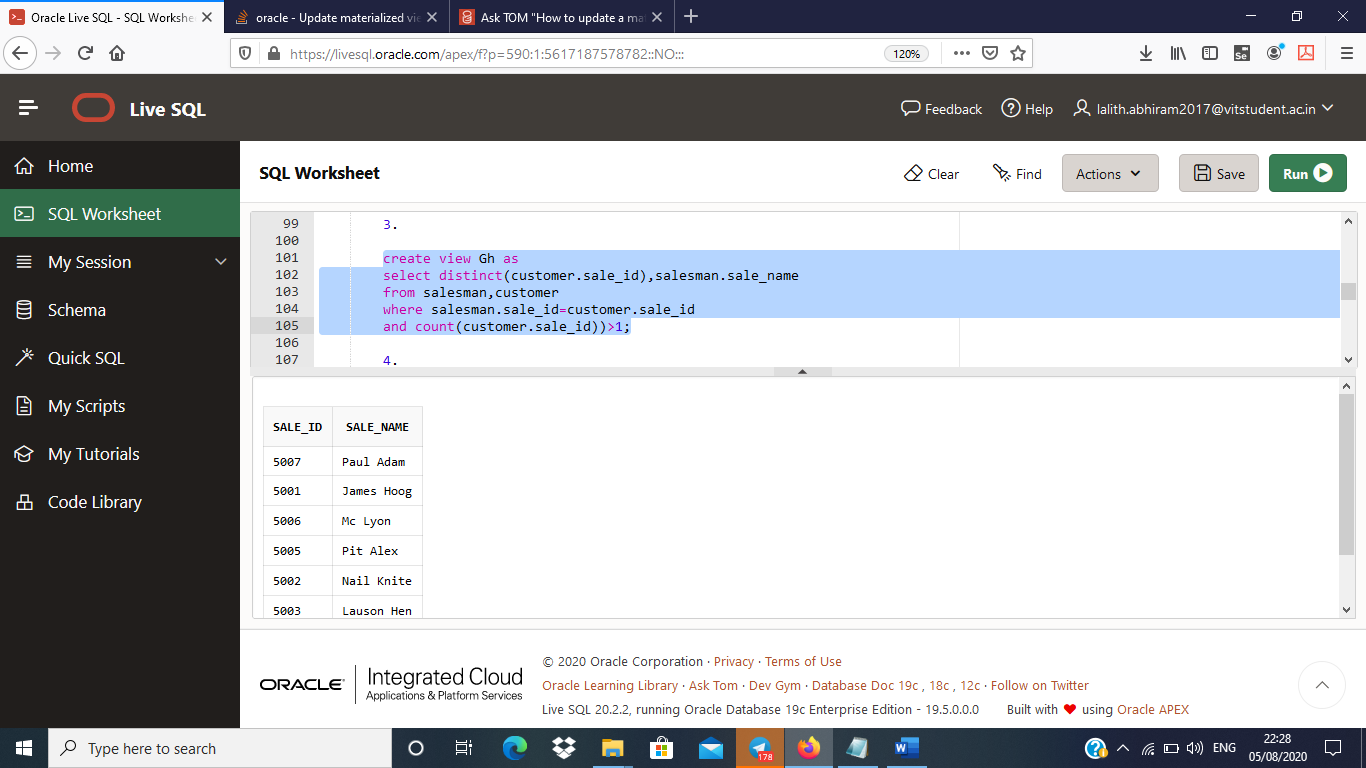
create view Gh as

select distinct(customer.sale\_id),salesman.sale\_name

from salesman,customer

where salesman.sale\_id=customer.sale\_id

and count(customer.sale\_id))>1;



1. Create a materialized view MVIEW4 on demand for Orders table. Perform Update on order table for id 70013 purchase amount = 1232.1 and Refresh the MVIEW4. Check whether the change on order table is reflected on MVIEW table.

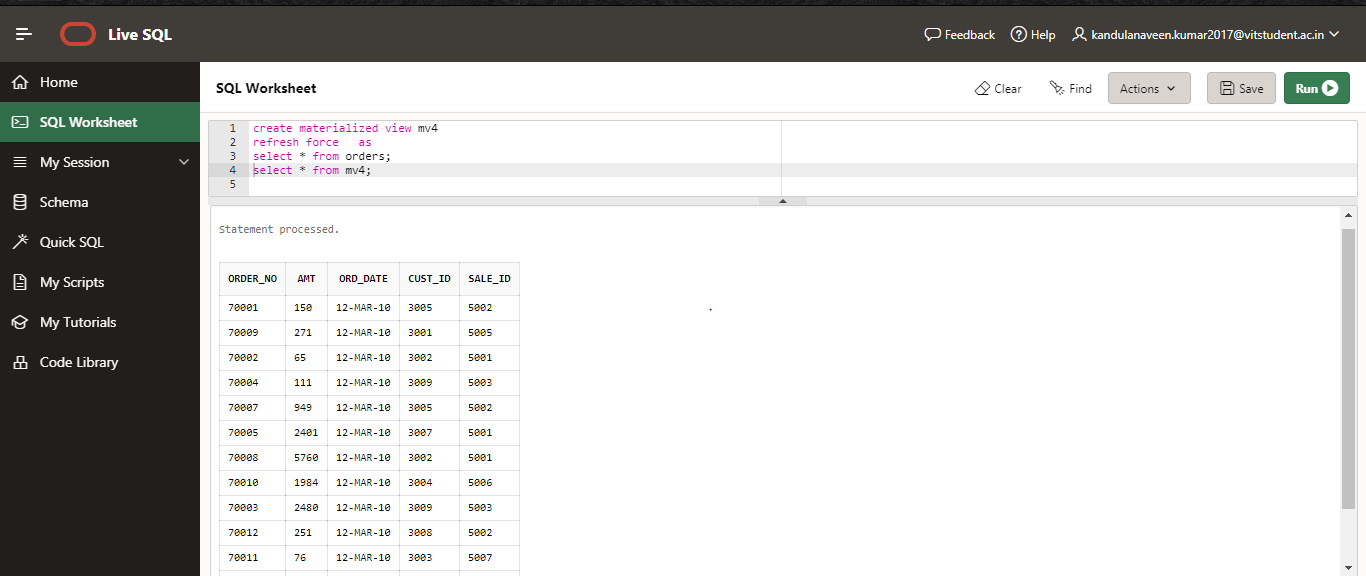
ANS:

create materialized view mv4

refresh force as

select \* from orders;

select \* from mv4;



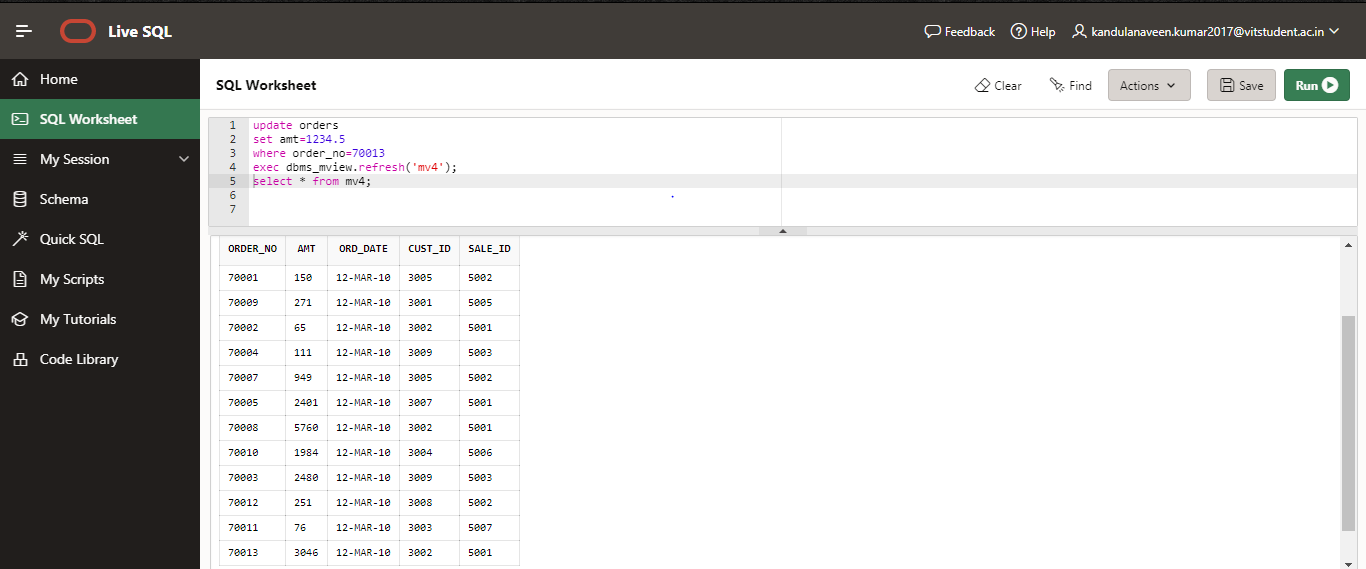
update orders

set amt=1234.5

where order\_no=70013

exec dbms\_mview.refresh('mv4');

select \* from mv4;



1. Create a materialized view MVIEW5 on commit for Salesman table, that finds the salesman who has the maximum commission. Perform a update on Salesman (Commission column) table and Refresh the MVIEW5. Check whether the change on Salesman table is reflected on MVIEW table.

ANS:

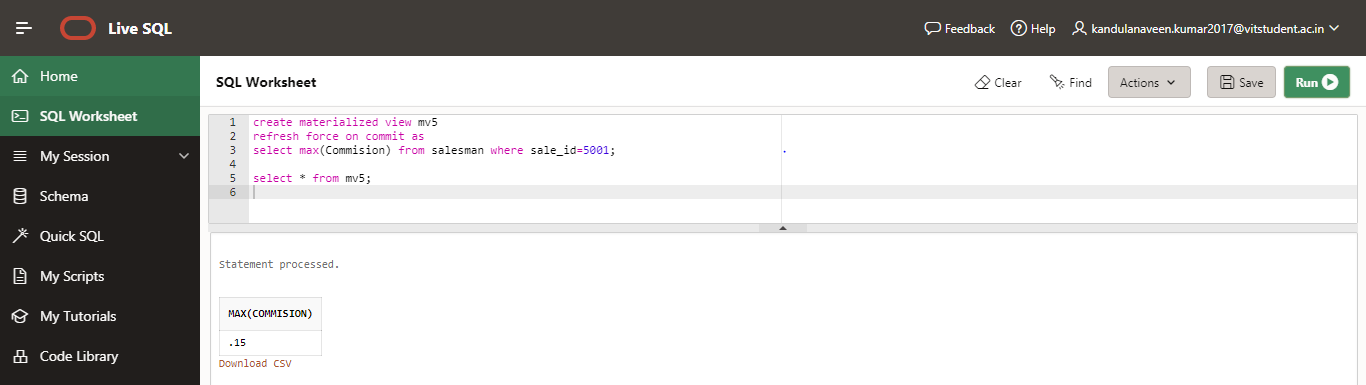
create materialized view mv5

refresh force

on commit as

select max(Commision) from salesman where sale\_id=5001;

select \* from mv5;



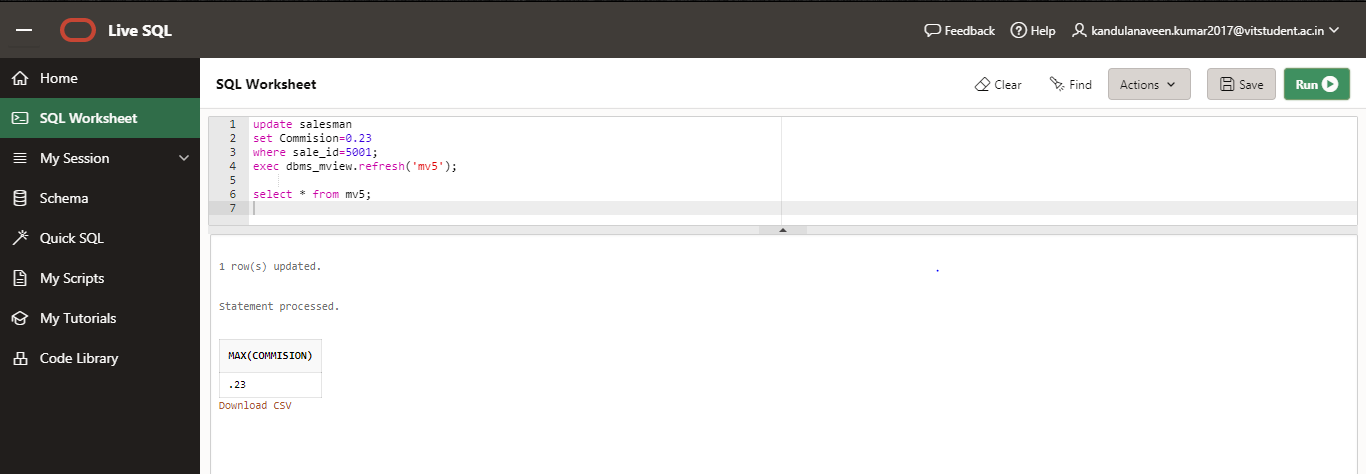
update salesman

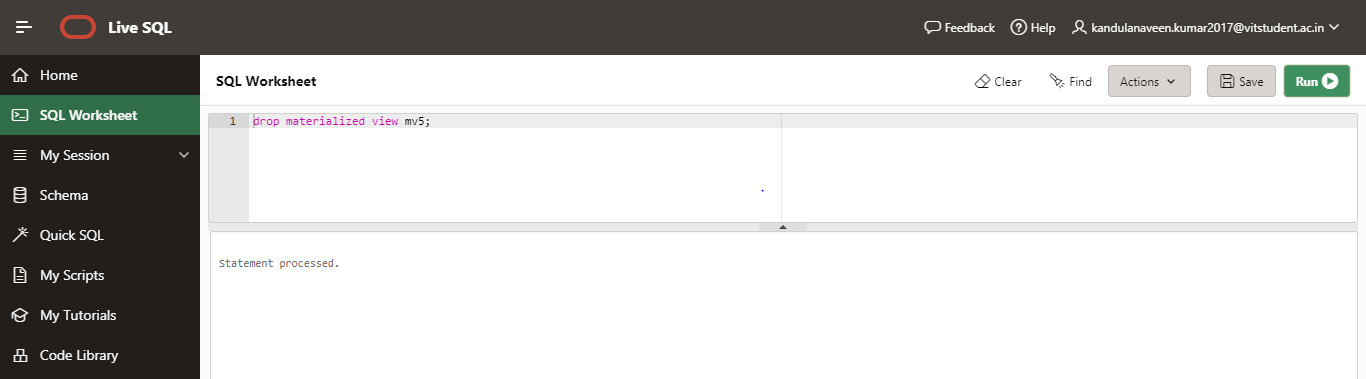
set Commision=0.23

where sale\_id=5001;

exec dbms\_mview.refresh('mv5');

select \* from mv5;





1. Create a materialized view that Display all the orders from the orders table issued by the salesman 'Paul Adam'. Perform (any) update on orders table and Fast Refresh the MVIEW. Check whether the change on Salesman table is reflected on MVIEW table.

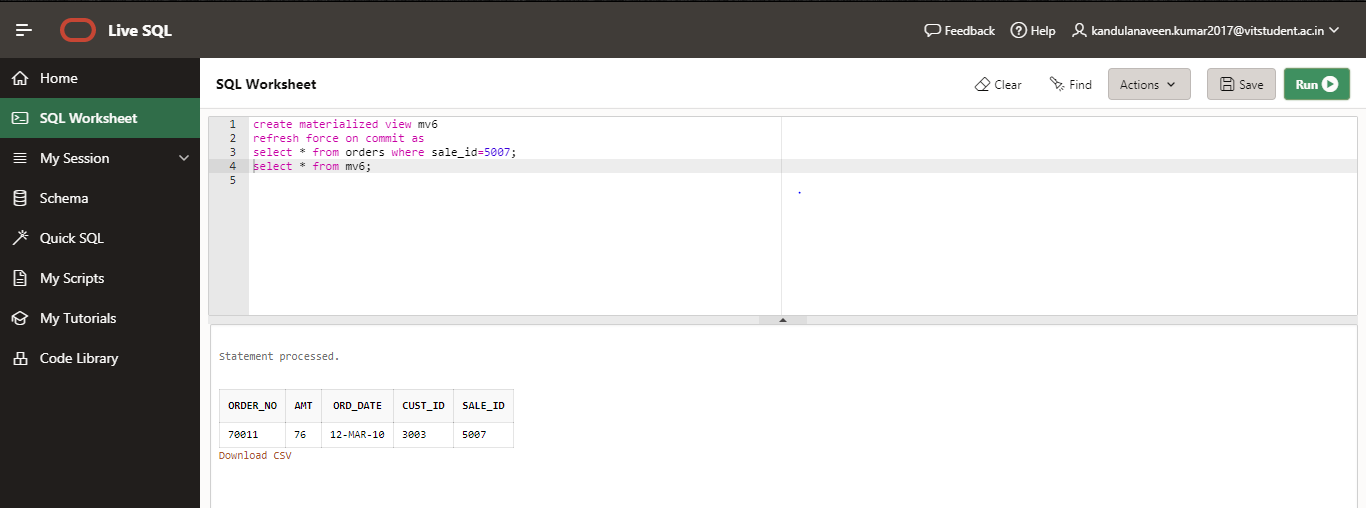
ANS:

create materialized view mv6

refresh force on commit as

select \* from orders where sale\_id=5007;

select \* from mv6;



update orders

set amt=85 where sale\_id=5007;

exec dbms\_mview.refresh('mv6');

select \* from mv6;

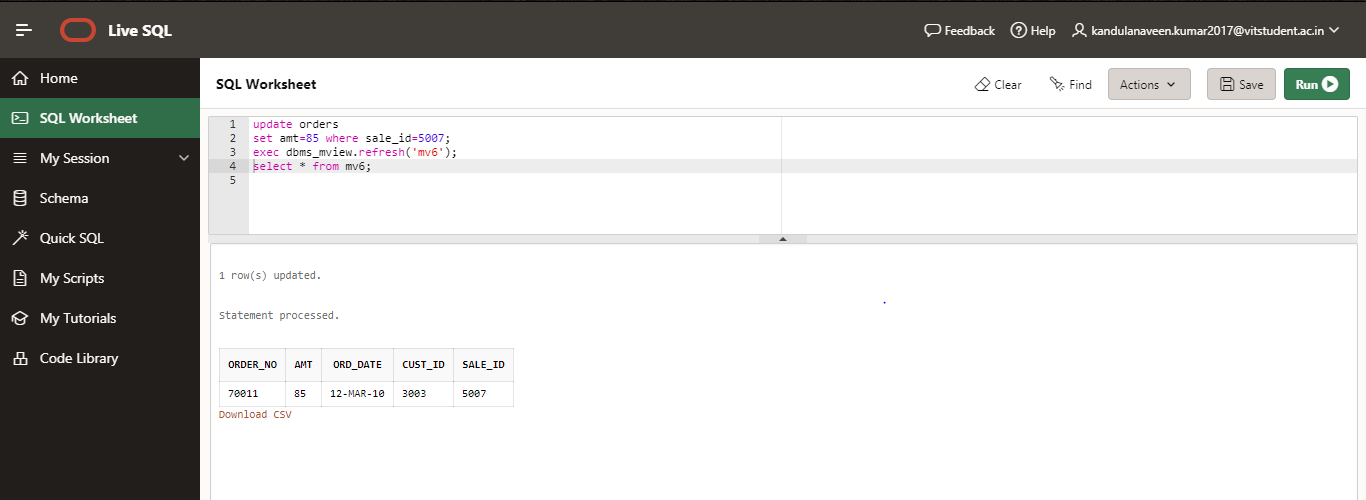


Table Creation

CREATE TABLE SALESMAN (SALESMAN\_ID NUMBER (4), NAME VARCHAR2 (20), CITY VARCHAR2 (20), COMMISSION VARCHAR2 (20), PRIMARY KEY (SALESMAN\_ID));

CREATE TABLE CUSTOMER (CUSTOMER\_ID NUMBER (4), CUST\_NAME VARCHAR2 (20), CITY VARCHAR2 (20), GRADE NUMBER (3), PRIMARY KEY (CUSTOMER\_ID), SALESMAN\_ID REFERENCES SALESMAN (SALESMAN\_ID) ON DELETE SET NULL);

CREATE TABLE ORDERS (ORD\_NO NUMBER (5), PURCHASE\_AMT NUMBER (10, 2), ORD\_DATE DATE, PRIMARY KEY (ORD\_NO), CUSTOMER\_ID REFERENCES CUSTOMER1 (CUSTOMER\_ID) ON DELETE CASCADE, SALESMAN\_ID REFERENCES SALESMAN (SALESMAN\_ID) ON DELETE CASCADE);

Grant the privileges:

grant GLOBAL QUERY REWRITE to scott;

grant CREATE TABLE to scott;

grant CREATE MATERIALIZED VIEW to scott;

Verify your users privileges:

connect scott/tiger

select \* from user\_sys\_privs;

USERNAME PRIVILEGE ADM

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

EKSTERN CREATE SESSION NO

EKSTERN CREATE TABLE NO

EKSTERN CREATE MATERIALIZED VIEW NO

EKSTERN GLOBAL QUERY REWRITE NO