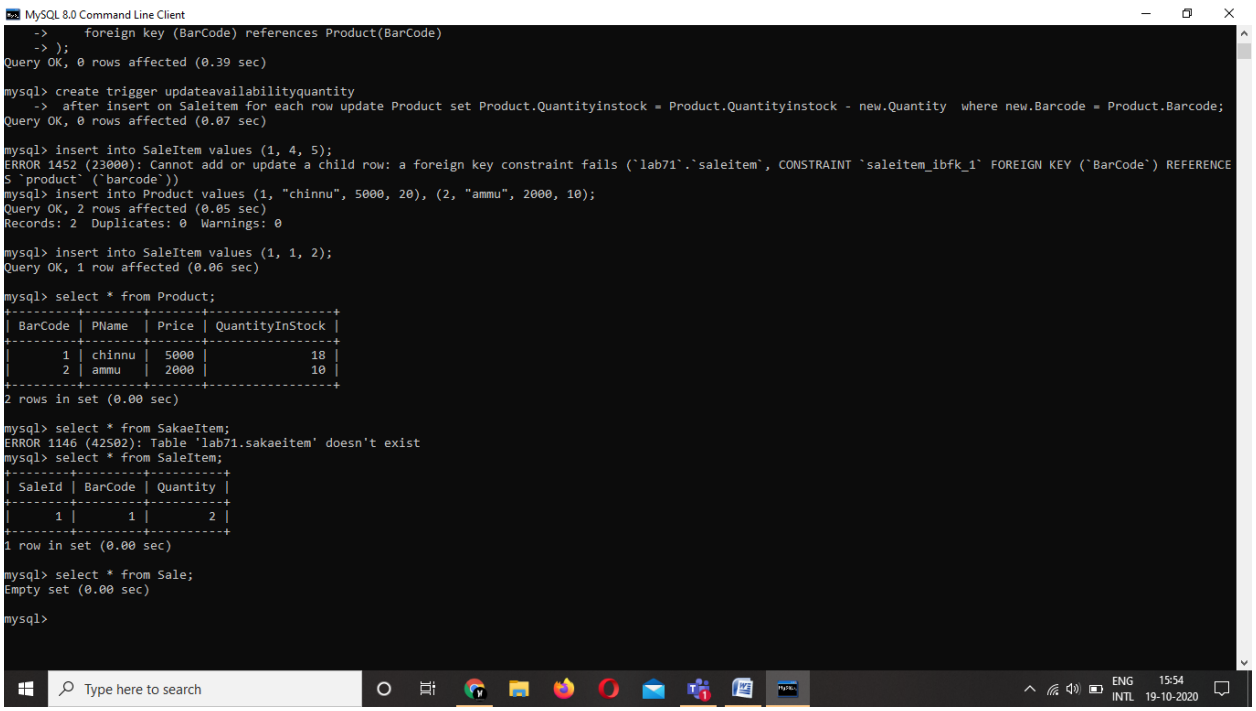
LAB 7:

1. Product (BarCode, PName, Price, QuantityInStock)
Sale (SaleID, DeliveryAddress, CreditCard)
SaleItem (SaleID, BarCode, Quantity)
Create a trigger called updateAvailableQuantity that updates the quantity in stock in the Product table, for every product sold. The trigger should be executed after each insert operation on the SaleItem table: for the product with the given barcode (the one inserted into SaleItem), update the available quantity in Product table to be the old quantity minus the sold quantity.

```
create table Product (
   BarCode int,
   PName varchar(40),
   Price int,
   QuantityInStock int,
   primary key (BarCode));

create table Sale (
   SaleId int,
   DeliveryAddress varchar(40),
   CreditCard int,
   primary key(SaleID));

create table SaleItem (
   SaleId int,
   BarCode int,
   Quantity int,
   primary key (BarCode, SaleId),
   foreign key (BarCode) references Product(BarCode));
create trigger updateavailabilityquantity

 after insert on Saleitem for each row update Product set Product.Quantityinstock =
Product.Quantityinstock - new.Quantity  where new.Barcode = Product.Barcode;

insert into Product values (1, "chinnu", 5000, 20), (2, "ammu", 2000, 10);

insert into SaleItem values (1, 1, 2);
```

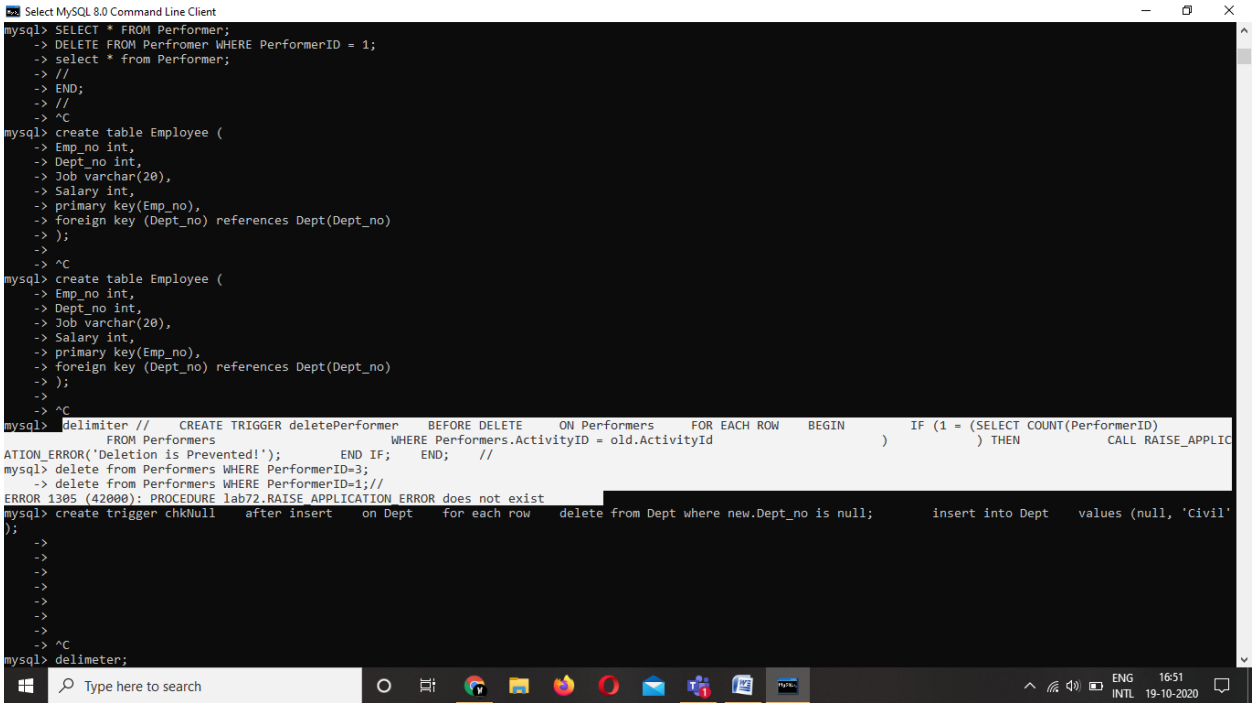


2. Activities (ActivityID, ActivityName)
Performers (PerformerID, PerformerName, Street, City, State, Zip, ActivityID)
Arenas (ArenaID, ArenaName, City, ArenaCapacity)
Concerts (PerformerID, ArenaID, ConcertDate, TicketPrice)
Create a trigger called deletePerformer that prevents the last performer for a particular activity to be deleted from the database. The trigger should be associated with a delete operation on the Peformers table: if the performer to be deleted is the last one for his/her particular ActivityID then the performer cannot be deleted. Otherwise, the performer should be deleted.

```
create table Activites (
   ActivityID int not null,
    ActivityName varchar(40),
    primary key (ActivityID)
    );
create table Performers (
   PerformerID  int not null,
  PerformerName varchar(40),
    Street varchar(40),
```

```
   City varchar(40),
State varchar(40),
Zip int(6),
     ActivityID int,
   primary key (PerformerID),
foreign key (ActivityID) references Activites(ActivityID)
   );
create table Arenas (
     ArenaID int not null,
   ArenaName varchar(40),
City varchar(40),
   ArenaCapacity varchar(40),
primary key (ArenaID)
   );
create table Concerts (
   PerformerID int,
   ArenaID int,
   ConcertDate datetime,
   TicketPrice int,
     primary key (PerformerID, ArenaID),
foreign key (PerformerID) references Performers(PerformerID),
     foreign key (ArenaID) references Arenas(ArenaID)
   );
insert into Activites
values (1, "sports"), (2, "Swimming"),(3, "singing");
```

```
insert into Performers
values (1, "priyanka ", "Thuvakudi", "Trichy", "Tamil Nadu", 620015, 1),
        (2, "bharath", "Thuvakudi", "Trichy", "Tamil Nadu", 620015, 1),
        (3, "chinnu", "Thuvakudi", "Trichy", "Tamil Nadu", 620015, 2);
```



3. Employee(Emp_no, Dept_no, Job, Salary) Dept(Dept_no, Dept_name)

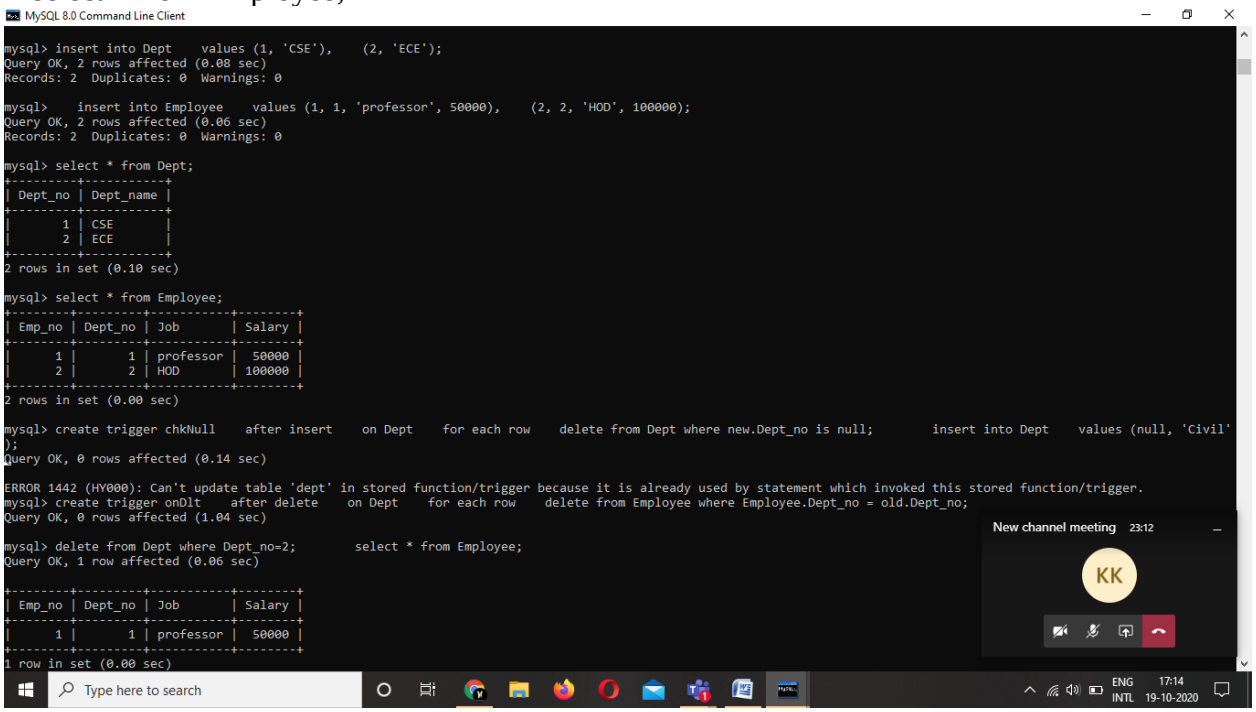
```
create table Employee (
        Emp_no int primary key,
        Dept_no int,
        Job varchar(20),
        Salary int
   );

   create table Dept (
        Dept_no int,
        Dept_name varchar(20)
   );

   insert into Dept
   values (1, 'CSE'),
        (2, 'ECE');

   insert into Employee
   values (1, 1, 'professor', 50000),
        (2, 2, 'HOD', 100000);
```

1. Write a TRIGGER to ensure that DEPT TABLE does not contain duplicate of null values in DEPTNO column.

```
create trigger one
    before insert
    on Dept
    for each row
    begin
    if ( new.Dept_no is null)
    then
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = "null not allowed";
    end if;
    end;
    $$
```

2. Write a Trigger to carry out the following action: on deleting a deptno from dept table , all the records with that deptno has to be deleted from the emp table

```
create trigger onDlt
    after delete
    on Dept
    for each row
    delete from Employee where Employee.Dept_no = old.Dept_no;


    delete from Dept where Dept_no=2;

    select * from Employee;
```



3. Write a trigger that raises an User defined error message and does not allow updating and insertion.

```
create trigger insUpd
```

```
  before insert
  on Dept
  for each row
  CALL RAISE_APPLICATION_ERR0R('auth error');
insert into Dept
  values (3, 'EEE');
```