Name: Muhammad Uzair

Roll No. L1F18BSCS0311

Section-H

Database

Lab 15 Part 1

Triggers

Q.1

(a)

create table customer (

acc\_no int primary key,

cust\_name varchar(20),

avail\_balance decimal);

create table mini\_statement (

last\_update datetime,

acc\_no int,

avail\_balance decimal,

foreign key(acc\_no) references customer(acc\_no) on delete cascade);

insert into customer values (1000, "Fanny", 7000);

insert into customer values (1001, "Peter", 12000);

delimiter //

create trigger update\_cus

before update

on

customer

for each row

begin

insert into mini\_statement values (CURRENT\_TIMESTAMP(),old.acc\_no,old.avail\_balance);

end;

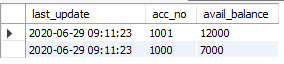
//

Delimiter ;

update customer set avail\_balance = avail\_balance + 3000 where acc\_no = 1001;

update customer set avail\_balance = avail\_balance + 3000 where acc\_no = 1000;

select \* from mini\_statement;



(b)

create table micro\_statement (

last\_update datetime,

acc\_no int,

avail\_balance decimal,

foreign key(acc\_no) references customer(acc\_no) on delete cascade);

insert into customer values (1002, "Janitor", 4500);

delimiter //

create trigger update\_after

after update

on

customer

for each row

begin

insert into micro\_statement values(CURRENT\_TIMESTAMP(), new.acc\_no,new.avail\_balance);

end;

//

delimiter ;

update customer set avail\_balance = avail\_balance + 1500 where acc\_no = 1002;

select \* from micro\_statement;



(c)

The difference between Before Update and After Update is that in before update, the trigger enacts before an update is invoked. If an update statement is written, this trigger will perform its actions before the update is implemented. In after update, the tigger enacts after an update is invoked. If an update statement is writeen, this trigger will perform its actions after that update statement has been executed.

Q.2

(a)

create table contacts (

contact\_id INT (11) NOT NULL AUTO\_INCREMENT,

last\_name VARCHAR (20) NOT NULL,

first\_name VARCHAR (25),

birthday DATE,

created\_date DATE,

created\_by VARCHAR(20),

CONSTRAINT contacts\_pk PRIMARY KEY (contact\_id));

delimiter //

create trigger contacts\_before\_insert

before insert

on

contacts

for each row

begin

DECLARE vUser varchar(50);

select USER() into vUser;

SET NEW.created\_date = SYSDATE();

SET NEW.created\_by = vUser;

end;

//

Delimiter ;

insert into contacts values (1, "Newton", "Enigma",str\_to\_date ("19-08-1999", "%d-%m-%Y"),str\_to\_date ("17-03-2018", "%d-%m-%Y"), "xyz");

select \* from contacts;



(b)

create table contacts1 (

contact\_id int (11) NOT NULL AUTO\_INCREMENT,

last\_name VARCHAR(30) NOT NULL,

first\_name VARCHAR(25),

birthday DATE,

CONSTRAINT contacts\_pk PRIMARY KEY (contact\_id));

create table contacts1\_audit (

contact\_id integer,

created\_date date,

created\_by varchar (30));

delimiter //

create trigger contacts\_after\_insert

after insert

on

contacts1

for each row

begin

DECLARE vUser varchar(30);

SELECT USER() into vUser;

INSERT into contacts1\_audit ( contact\_id, created\_date, created\_by)

VALUES ( NEW.contact\_id, SYSDATE(), vUser );

END;

//

insert into contacts1 values (1, "Asif", "Majeed",str\_to\_date("20-06-1999", "%d-%m-%Y"));

select \* from contacts1\_audit;



(c)

The difference between Before Insert and After Insert is that in before insert, the trigger is invoked before an insert statement is executed. This means that whenever we execute an insert statement, this trigger will perform its action before its execution. In after insert, the trigger is invoked after an insert statement has been executed. Whenever we right an insert statement, this trigger will be invoked after it has been executed.

Q.3

(a)

create table contacts (

contact\_id int (11) NOT NULL AUTO\_INCREMENT,

last\_name VARCHAR (30) NOT NULL,

first\_name VARCHAR (25),

birthday DATE,

created\_date DATE,

created\_by VARCHAR(30),

CONSTRAINT contacts\_pk PRIMARY KEY (contact\_id));

create table contacts\_audit (

contact\_id integer,

deleted\_date date,

deleted\_by varchar(20));

delimiter //

create trigger contacts\_before\_delete

before delete

on

contacts

for each row

begin

DECLARE vUser varchar(50);

SELECT USER() into vUser;

INSERT into contacts\_audit ( contact\_id, deleted\_date, deleted\_by)

VALUES ( OLD.contact\_id, SYSDATE(), vUser );

end;

//

insert into contacts values (1, "Bond", "Ruskin",str\_to\_date ("19-08-1995", "%d-%m-%Y"),str\_to\_date ("27-04-2018", "%d-%m-%Y"), "xyz");

delete from contacts where last\_name = "Bond";

select \*from contacts\_audit;



(b)

create table contacts (

contact\_id int (11) NOT NULL AUTO\_INCREMENT,

last\_name VARCHAR (30) NOT NULL,

first\_name VARCHAR (25),

birthday DATE,

created\_date DATE,

created\_by VARCHAR (30),

CONSTRAINT contacts\_pk PRIMARY KEY (contact\_id));

create table contacts\_audit (

contact\_id integer,

deleted\_date date,

deleted\_by varchar(20));

delimiter //

create trigger contacts\_after\_delete

after delete

on contacts

for each row

begin

DECLARE vUser varchar(50);

SELECT USER() into vUser;

INSERT into contacts\_audit ( contact\_id, deleted\_date, deleted\_by)

VALUES ( OLD.contact\_id, SYSDATE(), vUser );

end;

//

insert into contacts values (1, "Newton", "Isaac",str\_to\_date ("19-08-1985", "%d-%m-%Y"),str\_to\_date ("23-07-2018", "%d-%m-%Y"), "xyz");

delete from contacts where first\_name="Isaac";

select \*from contacts\_audit;



(c)

The difference between before delete and after delete is that in before delete, the trigger is invoked before a delete statement is implement. It performs its actions before the execution of a delete statement. In after delete, the trigger is invoked after a delete statement is implemented. It performs its actions after the execution of a delete statement.

Lab 15 Part 2

Views

Q.1

create view Sale as

select upper(employees.first\_name), lower(employees.last\_name)

from employees

inner join orders

on employees.id = orders.employee\_id

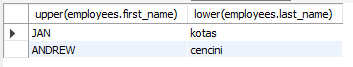
inner join order\_details

on orders.id = order\_details.order\_id

inner join products

on order\_details.product\_id = products.id and products.product\_name like '%Northwind Traders Dried Pears%';

select \* from Sale;



Q.2

Alter view Sale as

select employees.\*, products.product\_code

from employees

inner join orders

on employees.id = orders.employee\_id

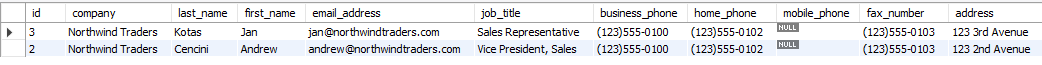
inner join order\_details

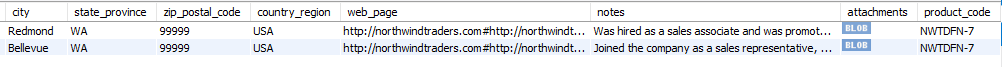
on orders.id = order\_details.order\_id

inner join products

on order\_details.product\_id = products.id and products.product\_name like '%Northwind Traders Dried Pears%';

select \* from Sale;





Q.3

create view Customer\_order as

select concat(first\_name,' ',last\_name) as Full\_Name,

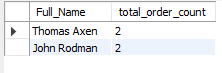
(select count(customer\_id) from orders where orders.customer\_id = customers.id and payment\_type = 'Cash') as total\_order\_count

from customers

group by (id)

having total\_order\_count > 0;

select \* from Customer\_order;



Q.4

Alter view Customer\_order as

select concat(first\_name,' ',last\_name) as Full\_Name,

(select count(customer\_id) from orders where orders.customer\_id = customers.id and payment\_type = 'Check') as total\_order\_count

from customers

group by (id)

having total\_order\_count >= 0;

select \* from Customer\_order;





Q.5

create view Orders\_view as

select id, order\_date, (select status\_name from orders\_status where orders\_status.id = status\_id) as stats

from orders where exists

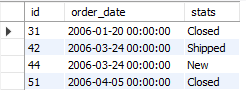
(select count(order\_details.order\_id) as different

from order\_details

where orders.id = order\_id

having different >= 3);

select \* from Orders\_view;



Q.6

Drop view Orders\_view;



Q.7

create view Employee\_purchase as

select concat(first\_name,' ',last\_name) as Employee\_Name

from employees where id IN

(select employee\_id from employee\_privileges

where privilege\_id IN

(select id from privileges where privilege\_name = 'Purchase Approvals'));

select \* from Employee\_purchase;



Q.8

Alter view Employee\_purchase as

select concat(first\_name,' ',last\_name) as Employee\_Name

from employees where id IN

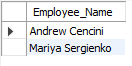
(select employee\_id from employee\_privileges

where privilege\_id IN

(select id from privileges where privilege\_name = 'Purchase Approvals'))

and employees.city = 'Redmond' or employees.city = 'Bellevue' or employees.city = 'kirkland';

select \* from Employee\_purchase;



Q.9

create view Products\_view as

select product\_name, sum(unit\_cost) as Amount

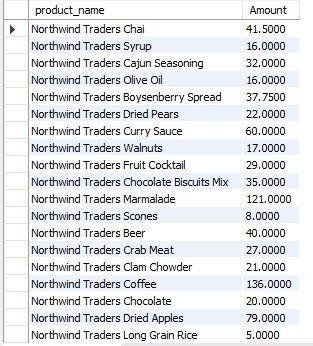
from products

inner join purchase\_order\_details

on products.id = product\_id

group by(product\_id);

select \* from Products\_view;





Q.10

ALter view Products\_view as

select product\_name, sum(unit\_cost) as Amount

from products

left join purchase\_order\_details

on products.id = product\_id

group by(product\_id);

select \* from Products\_view;

