**Tables Creation**

1.

CREATE TABLE "LWOS"."COMMENT\_TAB"

( "COMMENT\_ID" NUMBER(\*,0) NOT NULL ENABLE,

"COMMENT\_CONTENT" VARCHAR2(500 BYTE),

"COMMENT\_POST\_TYPE" VARCHAR2(20 BYTE),

"COM\_USER\_ID" NUMBER(\*,0),

"POST\_ID" NUMBER(\*,0),

"TIME\_DATE" VARCHAR2(100 BYTE),

"USER\_NAME\_COM" VARCHAR2(30 BYTE),

CONSTRAINT "COMMENT\_TAB\_PK" PRIMARY KEY ("COMMENT\_ID")

2.

CREATE TABLE "LWOS"."FOLLOW"

( "FOLLOW\_ID" NUMBER(\*,0) NOT NULL ENABLE,

"FOLLOWER\_USER\_ID" NUMBER(\*,0) NOT NULL ENABLE,

"FOLLOWING\_USER\_ID" NUMBER(\*,0) NOT NULL ENABLE,

"IS\_ACTIVE" NUMBER DEFAULT NULL NOT NULL ENABLE

)

3.

CREATE TABLE "LWOS"."NOTIFICATION"

( "NOTIFICATION\_ID" NUMBER(\*,0),

"FOLLOWER\_ID" NUMBER(\*,0),

"FOLLOWING\_ID" NUMBER(\*,0),

"NOTIFICATION\_MES" VARCHAR2(100 BYTE),

"IS\_NEW\_FLAG" NUMBER(\*,0),

"TIME\_DATE" VARCHAR2(60 BYTE)

)

4.

CREATE TABLE "LWOS"."POST\_TAB"

( "POST\_ID" NUMBER(\*,0),

"POST" VARCHAR2(3999 BYTE),

"POST\_TYPE" VARCHAR2(20 BYTE),

"CATEGORIES" VARCHAR2(20 BYTE),

"POST\_HEADLINE" VARCHAR2(50 BYTE),

"DATE\_TIME" VARCHAR2(60 BYTE),

"USER\_ID" NUMBER(\*,0),

"USER\_NAME\_POST" VARCHAR2(30 BYTE),

"POST\_NUMBER" NUMBER(\*,0),

"POST\_PHOTO" VARCHAR2(100 BYTE),

CONSTRAINT "DATA\_TAB\_PK" PRIMARY KEY ("POST\_ID")

5.

CREATE TABLE "LWOS"."POST\_TAB\_AUDIT"

( "POST\_ID" NUMBER(\*,0),

"POST" VARCHAR2(3999 BYTE),

"POST\_TYPE" VARCHAR2(20 BYTE),

"CATEGORIES" VARCHAR2(20 BYTE),

"POST\_HEADLINE" VARCHAR2(50 BYTE),

"DATE\_TIME" VARCHAR2(60 BYTE),

"USER\_ID" NUMBER(\*,0),

"USER\_NAME\_POST" VARCHAR2(30 BYTE),

"POST\_NUMBER" NUMBER(\*,0),

"POST\_PHOTO" VARCHAR2(100 BYTE),

"ORIGINAL\_ROW\_ID" VARCHAR2(200 BYTE),

"ACTION\_TYPE" VARCHAR2(200 BYTE),

"ACTION\_DATE" TIMESTAMP (6),

"ACTION\_BY" NUMBER,

"IP\_ADDRESS" VARCHAR2(200 BYTE),

"OS\_USER" VARCHAR2(200 BYTE),

"TERMINAL\_NAME" VARCHAR2(200 BYTE),

"HOST\_NAME" VARCHAR2(200 BYTE),

"USER\_NAME\_AUD" VARCHAR2(200 BYTE),

"ORA\_SYSEVENT" VARCHAR2(200 BYTE)

)

6.

CREATE TABLE "LWOS"."USERINFO"

( "USER\_ID" NUMBER(\*,0) NOT NULL ENABLE,

"USER\_ROLE" VARCHAR2(20 BYTE),

"FULL\_NAME" VARCHAR2(20 BYTE) NOT NULL ENABLE,

"USER\_NAME" VARCHAR2(20 BYTE) NOT NULL ENABLE,

"EMAIL" VARCHAR2(20 BYTE) NOT NULL ENABLE,

"MOBILE" VARCHAR2(20 BYTE) NOT NULL ENABLE,

"DOB" DATE NOT NULL ENABLE,

"GENDER" VARCHAR2(10 BYTE) NOT NULL ENABLE,

"ADDRESS" VARCHAR2(50 BYTE) NOT NULL ENABLE,

"COUNTRY" VARCHAR2(30 BYTE) NOT NULL ENABLE,

"CITY" VARCHAR2(20 BYTE),

"PASS" VARCHAR2(20 BYTE) NOT NULL ENABLE,

"PRO\_PIC" VARCHAR2(500 BYTE),

"POST\_AMOUNT" NUMBER(\*,0),

CONSTRAINT "USERINFO\_PK" PRIMARY KEY ("USER\_ID")

7.

CREATE TABLE "LWOS"."USERINFO\_AUDIT"

( "USER\_ID" NUMBER(\*,0) NOT NULL ENABLE,

"USER\_ROLE" VARCHAR2(20 BYTE),

"FULL\_NAME" VARCHAR2(20 BYTE) NOT NULL ENABLE,

"USER\_NAME" VARCHAR2(20 BYTE) NOT NULL ENABLE,

"EMAIL" VARCHAR2(20 BYTE) NOT NULL ENABLE,

"MOBILE" VARCHAR2(20 BYTE) NOT NULL ENABLE,

"DOB" DATE NOT NULL ENABLE,

"GENDER" VARCHAR2(10 BYTE) NOT NULL ENABLE,

"ADDRESS" VARCHAR2(50 BYTE) NOT NULL ENABLE,

"COUNTRY" VARCHAR2(30 BYTE) NOT NULL ENABLE,

"CITY" VARCHAR2(20 BYTE),

"PASS" VARCHAR2(20 BYTE) NOT NULL ENABLE,

"PRO\_PIC" VARCHAR2(500 BYTE),

"POST\_AMOUNT" NUMBER(\*,0),

"ORIGINAL\_ROW\_ID" VARCHAR2(200 BYTE),

"ACTION\_TYPE" VARCHAR2(200 BYTE),

"ACTION\_DATE" TIMESTAMP (6),

"ACTION\_BY" NUMBER,

"IP\_ADDRESS" VARCHAR2(200 BYTE),

"OS\_USER" VARCHAR2(200 BYTE),

"TERMINAL\_NAME" VARCHAR2(200 BYTE),

"HOST\_NAME" VARCHAR2(200 BYTE),

"USER\_NAME\_AUD" VARCHAR2(200 BYTE),

"ORA\_SYSEVENT" VARCHAR2(200 BYTE)

)

**Procedure**

1.

create or replace PROCEDURE COMMENT\_PRO

(COM\_CON COMMENT\_TAB.COMMENT\_CONTENT%TYPE, COM\_PO\_TY COMMENT\_TAB.COMMENT\_POST\_TYPE%TYPE,

COM\_U\_ID COMMENT\_TAB.COM\_USER\_ID%TYPE , PO\_ID COMMENT\_TAB.POST\_ID%TYPE ,

US\_NA\_COM COMMENT\_TAB.USER\_NAME\_COM%TYPE) IS

BEGIN

INSERT INTO COMMENT\_TAB (COMMENT\_ID,COMMENT\_CONTENT,COMMENT\_POST\_TYPE,COM\_USER\_ID,POST\_ID,USER\_NAME\_COM)

VALUES (comment\_id.nextval,COM\_CON,COM\_PO\_TY,COM\_U\_ID , PO\_ID ,US\_NA\_COM );

COMMIT ;

END ;

2.

create or replace PROCEDURE DELETE\_POST (ID POST\_TAB.POST\_ID%TYPE ) IS

BEGIN

DELETE FROM POST\_TAB WHERE POST\_ID=ID ;

commit ;

END ;

3.

create or replace PROCEDURE INSERT\_POST (POSTT POST\_TAB.POST%TYPE ,P\_TYPE POST\_TAB.POST\_TYPE%TYPE,

CATA POST\_TAB.CATEGORIES%TYPE, POST\_H POST\_TAB.POST\_HEADLINE%TYPE, P\_USER\_ID POST\_TAB.USER\_ID%TYPE) is

BEGIN

INSERT INTO POST\_TAB (post\_id,post,post\_type,CATEGORIES,post\_headline,user\_id)

VALUES ( POST\_ID\_S.nextval, POSTT,P\_TYPE ,CATA ,POST\_H, P\_USER\_ID );

COMMIT;

END;

4.

create or replace PROCEDURE INSERT\_USER\_POST (POSTT POST\_TAB.POST%TYPE ,P\_TYPE POST\_TAB.POST\_TYPE%TYPE,

CATA POST\_TAB.CATEGORIES%TYPE,

POST\_H POST\_TAB.POST\_HEADLINE%TYPE, P\_USER\_ID POST\_TAB.USER\_ID%TYPE, U\_NAME POST\_TAB.USER\_NAME\_POST%TYPE ) is

BEGIN

INSERT INTO POST\_TAB (post\_id,post,post\_type,CATEGORIES,post\_headline,user\_id,USER\_NAME\_POST)

VALUES ( POST\_ID\_S.nextval, POSTT,P\_TYPE ,CATA ,POST\_H, P\_USER\_ID ,U\_NAME );

COMMIT;

END;

5.

create or replace PROCEDURE POST\_UPDATE (ID POST\_TAB.POST\_ID%TYPE, PPOST POST\_TAB.POST%TYPE,

POST\_HED POST\_TAB.POST\_HEADLINE%TYPE,CATE POST\_TAB.CATEGORIES%TYPE) IS

BEGIN

UPDATE POST\_TAB SET POST=PPOST, POST\_HEADLINE=POST\_HED,CATEGORIES=CATE

WHERE POST\_ID=ID ;

COMMIT ;

END;

6.

create or replace PROCEDURE PROFILE\_UPDATE (

ID USERINFO.USER\_ID%TYPE,

FN USERINFO.FULL\_NAME%TYPE,

UN USERINFO.USER\_NAME%TYPE,

EMA USERINFO.EMAIL%TYPE,

MOBA USERINFO.MOBILE%TYPE,

BD USERINFO.DOB%TYPE,

GEN USERINFO.GENDER%TYPE,

ADDR USERINFO.ADDRESS%TYPE,

PA USERINFO.PASS%TYPE ) IS

BEGIN

UPDATE USERINFO SET

FULL\_NAME=FN,

USER\_NAME=UN,

EMAIL=EMA,

MOBILE=MOBA,

DOB=BD,

GENDER=GEN,

ADDRESS=ADDR,

PASS=PA

WHERE

USER\_ID=ID ;

COMMIT ;

END ;

1.

create or replace TRIGGER post\_time\_update

BEFORE INSERT OR UPDATE ON POST\_TAB

FOR EACH ROW

BEGIN

:new.DATE\_TIME := TO\_CHAR(sysdate,'YY/MM/DD HH:MM:SS');

END;

2.

create or replace TRIGGER timeupdate

BEFORE INSERT OR UPDATE ON COMMENT\_TAB

FOR EACH ROW

BEGIN

:new.TIME\_DATE := TO\_CHAR(sysdate,'YY/MM/DD HH:MM:SS');

END;

3.

create or replace TRIGGEr TRG\_POST\_TAB\_audit

AFTER DELETE OR UPDATE

ON POST\_TAB

FOR EACH ROW

DECLARE

p\_IP\_ADDRESS VARCHAR2 (200);

p\_OS\_USER VARCHAR2 (200);

p\_TERMINAL\_NAME VARCHAR2 (200);

p\_HOST\_NAME VARCHAR2 (200);

p\_USER\_NAME VARCHAR2 (200);

p\_ORA\_SYSEVENT VARCHAR2 (200);

BEGIN

BEGIN

SELECT ora\_sysevent,

USER,

SYS\_CONTEXT ('userenv', 'ip\_address') ip\_add,

SYS\_CONTEXT ('USERENV', 'OS\_USER') os\_user,

SYS\_CONTEXT ('USERENV', 'TERMINAL') ter\_name,

SYS\_CONTEXT ('USERENV', 'HOST') host\_name

INTO p\_ORA\_SYSEVENT,

p\_USER\_NAME,

p\_IP\_ADDRESS,

p\_OS\_USER,

p\_TERMINAL\_NAME,

p\_HOST\_NAME

FROM DUAL;

EXCEPTION

WHEN OTHERS

THEN

NULL;

END;

BEGIN

IF UPDATING

THEN

INSERT INTO POST\_TAB\_AUDIT (

POST\_ID,

POST,

POST\_TYPE,

CATEGORIES,

POST\_HEADLINE,

DATE\_TIME,

USER\_ID,

USER\_NAME\_POST,

POST\_NUMBER,

POST\_PHOTO,

ORIGINAL\_ROW\_ID,

ACTION\_TYPE,

ACTION\_DATE,

ACTION\_BY,

IP\_ADDRESS,

OS\_USER,

TERMINAL\_NAME,

HOST\_NAME,

USER\_NAME\_aud,

ORA\_SYSEVENT)

VALUES (

:old.POST\_ID,

:old.POST,

:old.POST\_TYPE,

:old.CATEGORIES,

:old.POST\_HEADLINE,

:old.DATE\_TIME,

:old.USER\_ID,

:old.USER\_NAME\_POST,

:old.POST\_NUMBER,

:old.POST\_PHOTO,

:old.ROWID,

'UPDATED',

SYSDATE,

:new.USER\_ID,

p\_IP\_ADDRESS,

p\_OS\_USER,

p\_TERMINAL\_NAME,

p\_HOST\_NAME,

p\_USER\_NAME,

p\_ORA\_SYSEVENT);

ELSIF DELETING

THEN

INSERT INTO POST\_TAB\_AUDIT (

POST\_ID,

POST,

POST\_TYPE,

CATEGORIES,

POST\_HEADLINE,

DATE\_TIME,

USER\_ID,

USER\_NAME\_POST,

POST\_NUMBER,

POST\_PHOTO,

ORIGINAL\_ROW\_ID,

ACTION\_TYPE,

ACTION\_DATE,

ACTION\_BY,

IP\_ADDRESS,

OS\_USER,

TERMINAL\_NAME,

HOST\_NAME,

USER\_NAME\_aud,

ORA\_SYSEVENT)

VALUES (

:old.POST\_ID,

:old.POST,

:old.POST\_TYPE,

:old.CATEGORIES,

:old.POST\_HEADLINE,

:old.DATE\_TIME,

:old.USER\_ID,

:old.USER\_NAME\_POST,

:old.POST\_NUMBER,

:old.POST\_PHOTO,

:old.ROWID,

'DELETE',

SYSDATE,

:old.USER\_ID,

p\_IP\_ADDRESS,

p\_OS\_USER,

p\_TERMINAL\_NAME,

p\_HOST\_NAME,

p\_USER\_NAME,

p\_ORA\_SYSEVENT);

END IF;

EXCEPTION

WHEN OTHERS

THEN

NULL;

END;

END;

4.

create or replace TRIGGER TRG\_send\_notification

BEFORE INSERT ON POST\_TAB

FOR EACH ROW

-- declare

begin

begin

insert into notification (NOTIFICATION\_ID,FOLLOWER\_ID,FOLLOWING\_ID,NOTIFICATION\_MES,IS\_NEW\_FLAG,TIME\_DATE)

select seq\_notification.nextval,a.FOLLOWER\_USER\_ID,a.FOLLOWING\_USER\_ID, :new.post\_headline,1, to\_char(sysdate,'dd/mm/yyyy HH:MI:ss')

from follow a ,userinfo c

where c.user\_id=a.following\_user\_id

and a.FOLLOWING\_USER\_ID = :new.user\_id ;

exception

when others then

dbms\_output.put\_line(sqlerrm) ;

null ;

end ;

end ;

5.

create or replace TRIGGER TRG\_send\_notification\_commment

BEFORE INSERT ON COMMENT\_TAB

FOR EACH ROW

-- declare

begin

begin

insert into notification (NOTIFICATION\_ID,FOLLOWER\_ID,FOLLOWING\_ID,NOTIFICATION\_MES,IS\_NEW\_FLAG,TIME\_DATE)

select seq\_notification.nextval,a.FOLLOWER\_USER\_ID,a.FOLLOWING\_USER\_ID, :new.COMMENT\_CONTENT,1, to\_char(sysdate,'dd/mm/yyyy HH:MI:ss')

from follow a ,userinfo c

where c.user\_id=a.following\_user\_id

and a.FOLLOWING\_USER\_ID = :new.COM\_USER\_ID ;

exception

when others then

dbms\_output.put\_line(sqlerrm) ;

null ;

end ;

end ;

6.

create or replace TRIGGEr TRG\_userinfo\_audit

AFTER DELETE OR UPDATE

ON userinfo

FOR EACH ROW

DECLARE

p\_IP\_ADDRESS VARCHAR2 (200);

p\_OS\_USER VARCHAR2 (200);

p\_TERMINAL\_NAME VARCHAR2 (200);

p\_HOST\_NAME VARCHAR2 (200);

p\_USER\_NAME VARCHAR2 (200);

p\_ORA\_SYSEVENT VARCHAR2 (200);

BEGIN

BEGIN

SELECT ora\_sysevent,

USER,

SYS\_CONTEXT ('userenv', 'ip\_address') ip\_add,

SYS\_CONTEXT ('USERENV', 'OS\_USER') os\_user,

SYS\_CONTEXT ('USERENV', 'TERMINAL') ter\_name,

SYS\_CONTEXT ('USERENV', 'HOST') host\_name

INTO p\_ORA\_SYSEVENT,

p\_USER\_NAME,

p\_IP\_ADDRESS,

p\_OS\_USER,

p\_TERMINAL\_NAME,

p\_HOST\_NAME

FROM DUAL;

EXCEPTION

WHEN OTHERS

THEN

NULL;

END;

BEGIN

IF UPDATING

THEN

INSERT INTO userinfo\_audit (

USER\_ID,

USER\_ROLE,

FULL\_NAME,

USER\_NAME,

EMAIL,

MOBILE,

DOB,

GENDER,

ADDRESS,

COUNTRY,

CITY,

PASS,

PRO\_PIC,

POST\_AMOUNT,

ORIGINAL\_ROW\_ID,

ACTION\_TYPE,

ACTION\_DATE,

ACTION\_BY,

IP\_ADDRESS,

OS\_USER,

TERMINAL\_NAME,

HOST\_NAME,

USER\_NAME\_aud,

ORA\_SYSEVENT)

VALUES (

:old.USER\_ID,

:old.USER\_ROLE,

:old.FULL\_NAME,

:old.USER\_NAME,

:old.EMAIL,

:old.MOBILE,

:old.DOB,

:old.GENDER,

:old.ADDRESS,

:old.COUNTRY,

:old.CITY,

:old.PASS,

:old.PRO\_PIC,

:old.POST\_AMOUNT,

:old.ROWID,

'UPDATED',

SYSDATE,

:new.USER\_ID,

p\_IP\_ADDRESS,

p\_OS\_USER,

p\_TERMINAL\_NAME,

p\_HOST\_NAME,

p\_USER\_NAME,

p\_ORA\_SYSEVENT);

-- Trigger this when Deleting a Record

-- Insert data into Audit Table with Action Deleted

ELSIF DELETING

THEN

INSERT INTO userinfo\_audit (

USER\_ID,

USER\_ROLE,

FULL\_NAME,

USER\_NAME,

EMAIL,

MOBILE,

DOB,

GENDER,

ADDRESS,

COUNTRY,

CITY,

PASS,

PRO\_PIC,

POST\_AMOUNT,

ORIGINAL\_ROW\_ID,

ACTION\_TYPE,

ACTION\_DATE,

ACTION\_BY,

IP\_ADDRESS,

OS\_USER,

TERMINAL\_NAME,

HOST\_NAME,

USER\_NAME\_aud,

ORA\_SYSEVENT)

VALUES (

:old.USER\_ID,

:old.USER\_ROLE,

:old.FULL\_NAME,

:old.USER\_NAME,

:old.EMAIL,

:old.MOBILE,

:old.DOB,

:old.GENDER,

:old.ADDRESS,

:old.COUNTRY,

:old.CITY,

:old.PASS,

:old.PRO\_PIC,

:old.POST\_AMOUNT,

:old.ROWID,

'DELETE',

SYSDATE,

:old.USER\_ID,

p\_IP\_ADDRESS,

p\_OS\_USER,

p\_TERMINAL\_NAME,

p\_HOST\_NAME,

p\_USER\_NAME,

p\_ORA\_SYSEVENT);

END IF;

EXCEPTION

WHEN OTHERS

THEN

NULL;

END;

END;