**DataBase Final project**

**Sql queries**

**Group Member**

**21F-9214 Laiba Zafar**

**21F-9227 Saad Athar**

**21F-9132 Kamla safdar**

DROP TABLE ADMIN;

DROP TABLE CUSTOMER;

DROP TABLE VENDOR;

DROP TABLE SALESPERSON;

DROP TABLE PRODUCT;

DROP TABLE ORDERS;

DROP TABLE RACK;

DROP TABLE ORDER\_ITEMS;

DROP TABLE SALES;

DROP TABLE CATEGORY;

DROP TABLE TRANSACTION;

select \*from customer;

CREATE TABLE ADMIN (

username varchar(20) PRIMARY KEY,

Adminpassword varchar(20),

AdminName varchar(40) NOT NULL,

contactNo varchar(11) NOT NULL,

CNIC varchar(15) NOT NULL,

Designation VARCHAR(10) DEFAULT 'Admin'

);

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

CREATE TABLE SALES (

BILL\_ID INT NOT NULL primary key,

billPaidDate date null,

Total decimal(20),

paid decimal(20),

customerID INT NOT NULL --FK

);

drop table sales;

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

CREATE TABLE CART(

cartID INT PRIMARY KEY,

BILL\_ID INT, --FK

PRODUCT\_ID INT --FK

);

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

CREATE TABLE VENDOR1(

VID INT Primary key,

vName varchar2(20) NOT NULL,

CNIC varchar2(15),

contact varchar(11),

ActiveStatus INT --SOFT DELELTE ATTRIBUTE

);

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

CREATE TABLE ORDERS(

orderID INT NOT NULL PRIMARY KEY,

orderType varchar2(20) NOT NULL,

amount float NOT NULL,

orderDate DATE NOT NULL,

VID INT NOT NULL --FK

);

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

CREATE TABLE SALESPERSON(

SID INT NOT NULL PRIMARY KEY,

spName varchar(40) NOT NULL,

contact varchar(11) NOT NULL,

ActiveStatus INT --SOFT DELELTE ATTRIBUTE

);

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

CREATE TABLE RACK(

rackID INT NOT NULL PRIMARY KEY ,

floors INT NOT NULL,

capacity INT NOT NULL,

Placed INT DEFAULT(0),

ActiveStatus INT --SOFT DELELTE ATTRIBUTE

);

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

CREATE TABLE TRANSACTION(

transaction\_ID INT NOT NULL PRIMARY KEY,

transaction\_type varchar2(20) NOT NULL,

amount NUMBER(20) NOT NULL,

transactionDate DATE NOT NULL,

transactionTime TIMESTAMP NOT NULL,

BILL\_ID INT NOT NULL,

orderID INT NOT NULL

);

drop table transaction;

CREATE TABLE TRANSACTION(

transaction\_ID INT NOT NULL PRIMARY KEY,

transaction\_type varchar2(20) NOT NULL, --EXPENSE/REVENUE

amount NUMBER(20) NOT NULL,

transactionDate DATE NOT NULL,

BILL\_ID INT ,

orderID INT,

Active\_Status INT DEFAULT NULL --SOFT DELETE

);

SELECT SUM(CASE WHEN transaction\_type = 'REVENUE' THEN amount ELSE -amount END) AS Total\_Profit

FROM TRANSACTION;

select sum(amount)as Total\_Revenue

from transaction

where transaction\_type='revenue';

select sum(amount)as Total\_expense

from transaction

where transaction\_type='expense';

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

CREATE TABLE CATEGORY(

catName varchar2(20) NOT NULL PRIMARY KEY,

catID INT NOT NULL,

ActiveStatus INT --SOFT DELELTE ATTRIBUTE

);

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

CREATE TABLE CUSTOMER1(

customerID INT NOT NULL PRIMARY KEY,

customerName varchar2(20) NOT NULL,

contact varchar(11) NOT NULL,

CNIC varchar(15)

);

drop table customer1;

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

CREATE TABLE PRODUCT1 (

PRODUCT\_ID INT NOT NULL PRIMARY KEY,

PRODUCT\_NAME VARCHAR(50) NOT NULL,

PRODUCT\_WEIGHT FLOAT,

PRODUCT\_PRICE FLOAT,

BARCODE INT NOT NULL,

rackID INT NULL, --FK,

catName varchar2(20) NOT NULL, --FK

ActiveStatus INT --SOFT DELELTE ATTRIBUTE

);

select p.product\_id , p.product\_name , p.rackid , sum(r.placed) as item\_quantity

from product1 p , rack r

where p.rackid=r.rackid

group by p.product\_id , product\_name , p.rackid

having sum(r.placed)<10;

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

CREATE TABLE order\_items

(

orderID INT , -- fk

item\_id INT,

PRODUCT\_ID INT NOT NULL , -- fk

quantity NUMBER( 8, 2 ) NOT NULL ,

unit\_price NUMBER( 10, 2 ) NOT NULL ,

CONSTRAINT pk\_order\_items PRIMARY KEY( orderid, item\_id ), --composite primary key

CONSTRAINT fk\_order\_items\_products FOREIGN KEY( product\_id ) REFERENCES PRODUCT1 ( product\_id ) ON DELETE CASCADE,

CONSTRAINT fk\_order\_items\_orders FOREIGN KEY( orderid ) REFERENCES orders( orderid ) ON DELETE CASCADE

);

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

ALTER TABLE PRODUCT1

ADD CONSTRAINT rack\_fk FOREIGN KEY(rackID ) REFERENCES RACK(rackID ) ON DELETE CASCADE;

alter table product1

drop constraint rack\_fk;

ALTER TABLE PRODUCT1

ADD CONSTRAINT rack1\_fk FOREIGN KEY(rackID ) REFERENCES RACK(rackID) ON UPDATE CASCADE;

ALTER TABLE PRODUCT1 --product ki category kia hai

ADD CONSTRAINT category\_fk FOREIGN KEY(catName ) REFERENCES CATEGORY(catName ) ON DELETE CASCADE;

alter table product1

drop constraint category\_fk;

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

ALTER TABLE PRODUCT1

MODIFY CONSTRAINT update\_cascade REFERENCES RACK(rackID) ON UPDATE CASCADE;

alter table product1

drop constraint rack\_fk;

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

ALTER TABLE ORDERS --many orders can relate to a single vendor

ADD CONSTRAINT vendorID\_fk FOREIGN KEY(VID) REFERENCES VENDOR1(VID);

ALTER TABLE SALES --kis customer ka konsa bill hai( ERD-)

ADD CONSTRAINT customer\_fk FOREIGN KEY(customerID ) REFERENCES CUSTOMER1(customerID);

ALTER TABLE TRANSACTION

ADD CONSTRAINT bill\_fk FOREIGN KEY(BILL\_ID ) REFERENCES SALES(BILL\_ID );

ALTER TABLE CART

ADD CONSTRAINT bill\_id\_fk FOREIGN KEY(BILL\_ID) REFERENCES SALES(BILL\_ID );

ALTER TABLE CART

ADD CONSTRAINT product\_id\_fk FOREIGN KEY(PRODUCT\_ID ) REFERENCES PRODUCT1(PRODUCT\_ID );

ALTER TABLE TRANSACTION

ADD CONSTRAINT order\_fk FOREIGN KEY(orderID) REFERENCES ORDERS(orderID);

--customer,product,bill has a ternary relation. bill has customerID as FK in bill.

-- if I also add product\_id in bill tou for each product that a person has purchased,there will be a unique bill-ID. how is it possible?

///////////////////////////////////////////////////////////// --INSERTION CHECKS

ALTER TABLE ADMIN --username check

ADD CHECK (username LIKE '%@nu.edu.pk');

ALTER TABLE ADMIN --contact check

ADD CHECK (SUBSTR(ContactNo,1,4) BETWEEN '0000' AND '9999' AND SUBSTR(ContactNo,6) BETWEEN '0000000' AND '9999999');

ALTER TABLE ADMIN --CNIC check

ADD CHECK (LENGTH(CNIC) = 15 AND

SUBSTR(CNIC, 1, 5) BETWEEN '00000' AND '99999' AND

SUBSTR(CNIC, 6, 1) = '-' AND

SUBSTR(CNIC, 7, 7) BETWEEN '0000000' AND '9999999' AND

SUBSTR(CNIC, 14, 1) = '\_' AND

SUBSTR(CNIC, 15, 1) BETWEEN '0' AND '9');

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

ALTER TABLE VENDOR --contact check

ADD CHECK (SUBSTR(contact,1,4) BETWEEN '0000' AND '9999' AND SUBSTR(contact,6) BETWEEN '0000000' AND '9999999');

ALTER TABLE VENDOR --CNIC CHECK

ADD CHECK (LENGTH(CNIC) = 15 AND

SUBSTR(CNIC, 1, 5) BETWEEN '00000' AND '99999' AND

SUBSTR(CNIC, 6, 1) = '-' AND

SUBSTR(CNIC, 7, 7) BETWEEN '0000000' AND '9999999' AND

SUBSTR(CNIC, 14, 1) = '\_' AND

SUBSTR(CNIC, 15, 1) BETWEEN '0' AND '9');

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

CREATE SEQUENCE seq\_vendor --auto generate vendor ID

MINVALUE 1

START WITH 1

INCREMENT BY 12

CACHE 10;

ALTER TABLE CUSTOMER --contact check

ADD CHECK (SUBSTR(contact,1,4) BETWEEN '0000' AND '9999' AND SUBSTR(contact,6) BETWEEN '0000000' AND '9999999');

ALTER TABLE CUSTOMER --CNIC CHECK

ADD CHECK (LENGTH(CNIC) = 15 AND

SUBSTR(CNIC, 1, 5) BETWEEN '00000' AND '99999' AND

SUBSTR(CNIC, 6, 1) = '-' AND

SUBSTR(CNIC, 7, 7) BETWEEN '0000000' AND '9999999' AND

SUBSTR(CNIC, 14, 1) = '\_' AND

SUBSTR(CNIC, 15, 1) BETWEEN '0' AND '9');

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

CREATE SEQUENCE seq\_customer --auto generate customerID

MINVALUE 1

START WITH 1

INCREMENT BY 11

CACHE 10;

ALTER TABLE SALESPERSON --contact check

ADD CHECK (SUBSTR(contact,1,4) BETWEEN '0000' AND '9999' AND SUBSTR(contact,6) BETWEEN '0000000' AND '9999999');

CREATE SEQUENCE seq\_salesperson --auto-generate salespersonID

MINVALUE 1

START WITH 1

INCREMENT BY 4

CACHE 10;

CREATE SEQUENCE seq\_bill --auto-generate bill\_collectionID

MINVALUE 1

START WITH 1

INCREMENT BY 101

CACHE 10;

CREATE SEQUENCE seq\_order --auto-generate orderID

MINVALUE 1

START WITH 1

INCREMENT BY 1

CACHE 10;

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

ALTER TABLE TRANSACTION

ADD CHECK trans\_check(TO\_CHAR(transactionTime, 'HH12:MI:SS')) = transactionTime;

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

CREATE SEQUENCE seq\_transaction --auto-generate transactionID

MINVALUE 1

START WITH 1

INCREMENT BY 1

CACHE 10;

ALTER TABLE RACK --capacity check

ADD CONSTRAINT rack\_check CHECK (Placed<=capacity);

CREATE SEQUENCE seq\_rack --auto-generate rackID

MINVALUE 1

START WITH 1

INCREMENT BY 1

CACHE 10;

CREATE SEQUENCE seq\_cartid --auto-generate cart-ID

MINVALUE 1

START WITH 1

INCREMENT BY 1

CACHE 10;

CREATE SEQUENCE seq\_category --auto-generate categoryID

MINVALUE 1

START WITH 1

INCREMENT BY 1

CACHE 10;

CREATE SEQUENCE seq\_product --auto-generate product-ID

MINVALUE 56

START WITH 56

INCREMENT BY 11

CACHE 10;

CREATE SEQUENCE seq\_orderItem --auto-generate item-ID

MINVALUE 1

START WITH 1

INCREMENT BY 1

CACHE 10;

///////////////////////////////////////////////////////////////////////////// --not using this for now

CREATE OR REPLACE FUNCTION flex\_of\_Vendor(ID INT)

RETURN INT IS

temp\_data INT;

CURSOR tell\_cursor IS SELECT VID FROM VENDOR;

BEGIN

OPEN tell\_cursor;

LOOP

FETCH tell\_cursor INTO temp\_data;

EXIT WHEN tell\_cursor%NOTFOUND;

IF(ID = temp\_data) THEN

CLOSE tell\_cursor;

RETURN 1;

END IF;

END LOOP;

CLOSE tell\_cursor;

-- Return a default value if no match is found

RETURN 0;

END;

/

DECLARE

result INT:=0;

BEGIN

result := flex\_of\_Vendor(121);

DBMS\_OUTPUT.PUT\_LINE(result);

END;

/

/////////////////////////////////////////////////////////////////// --trigger to generate barcode

CREATE OR REPLACE TRIGGER GYM\_REQUEST\_INSERT\_TRG

BEFORE INSERT ON PRODUCT1

FOR EACH ROW

DECLARE

temp NUMBER;

BEGIN

temp := FLOOR(dbms\_random.value(100000000000, 999999999999));

:NEW.barcode := temp;

END;

drop trigger GYM\_REQUEST\_INSERT\_TRG;

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

/////////////////////////////////////////////////////////////////////////--trigger to increase PLACED IN RACK when insertion takes place in PRODUCT

create or replace trigger product\_insert\_trigg

AFTER INSERT ON PRODUCT

FOR EACH ROW

DECLARE

temp INT;

temp\_quantity INT;

BEGIN

select rackID INTO TEMP

from Product1 p

where p.product\_id= :NEW.product\_id;

select placed INTO temp\_quantity

from RACK

where rackID= temp;

temp\_quantity := temp\_quantity+1;

UPDATE RACK

SET PLACED= temp\_quantity where rackID=temp;

END;/

////////////////////////////////////////////////////////////////////// --trigger to decrease PLACED IN RACK when deletion takes place in PRODUCT

create or replace trigger product\_delete\_trigg

AFTER INSERT ON PRODUCT

FOR EACH ROW

DECLARE

temp INT;

temp\_quantity INT;

BEGIN

select p.rackID INTO TEMP

from Product1 p

where p.product\_id= :NEW.product\_id;

select placed INTO temp\_quantity

from RACK

where rackID= temp;

temp\_quantity := temp\_quantity-1;

UPDATE RACK

SET PLACED= temp\_quantity where rackID=temp;

END;/

drop trigger product\_delete\_trigg;

///////////////////////////////////////////////////////////////////////////// --trigger to decrease placed when insertion in cart

create or replace trigger cart\_insert\_trig

AFTER INSERT ON CART

FOR EACH ROW

DECLARE

temp INT;

temp\_quantity INT;

BEGIN

select p.rackID INTO TEMP

from Product1 p

where p.product\_id= :NEW.product\_id;

select placed INTO temp\_quantity

from RACK

where rackID= temp;

temp\_quantity := temp\_quantity - 1;

UPDATE RACK

SET PLACED= temp\_quantity where rackID=temp;

END;/

//////////////////////////////////////////////////////////////////// --trigger to increase placed placed when deletionin cart

create or replace trigger cart\_delete\_trig

AFTER DELETE ON CART

FOR EACH ROW

DECLARE

temp INT;

temp\_quantity INT;

BEGIN

select p.rackID INTO TEMP

from Product1 p

where p.product\_id= :NEW.product\_id;

select placed INTO temp\_quantity

from RACK

where rackID= temp;

temp\_quantity := temp\_quantity + 1;

UPDATE RACK

SET PLACED= temp\_quantity where rackID=temp;

END;/