Exercise - 1

Consider the following relations for a bus reservation system application:
BUS (ROUTENO, SOURCE, DESTINATION)

```
CREATE TABLE BUS_T
(
   ROUTENO  INT  NOT NULL,
   SOURCE   VARCHAR(50) NULL,
   DESTINATION VARCHAR(50) NULL,
   PRIMARY KEY(ROUTENO)
)
```

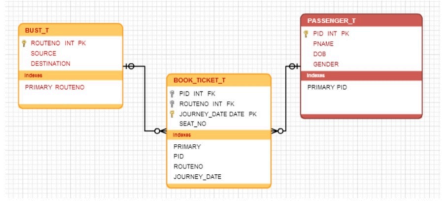
PASSENGER (PID, PNAME, DOB, GENDER)

```
CREATE TABLE PASSENGER_T
(
   PID  INT  NOT NULL,
   PNAME   VARCHAR (50) NULL,
   DOB   DATE NULL,
   GENDER VARCHAR (50) NULL,
    CONSTRAINT CHECK_PASS_PRI PRIMARY KEY(PID),
    CONSTRAINT CHECK_PASS_DOB check (EXTRACT(YEAR FROM DOB) > 2010
   )
```

BOOK_TICKET (PID, ROUTENO, JOURNEY_DATE, SEAT_NO)

```
CREATE TABLE BOOK_TICKET_T
(
   PID  INT NOT NULL,
   ROUTENO  INT NOT NULL,
   JOURNEY_DATE   DATE NOT NULL,
   SEAT_NO INT NULL,
   PRIMARY KEY(PID,ROUTENO,JOURNEY_DATE)
)
```

a. The primary keys are underlined. Identify the foreign keys and draw schema diagram [5]



b. Create the above mentioned tables and populate the tables [15]

```
INSERT INTO BUS_T VALUES ( 101,'PUNE','CHENNAI' );
INSERT INTO BUS_T VALUES ( 102,'PUNE','BANGLORE' );
INSERT INTO BUS_T VALUES ( 103,'CHENNAI','PUNE' );
INSERT INTO BUS_T VALUES ( 104,'CHENNAI','BANGLORE' );
INSERT INTO BUS_T VALUES ( 105,'BANGLORE','PUNE' );
INSERT INTO BUS_T VALUES ( 106,'BANGLORE','PUNE' );

INSERT INTO PASSENGER_T VALUES ( 201,'Ashok','10/10/2011','M' );
INSERT INTO PASSENGER_T VALUES ( 202,'Banty','10/10/2015','M' );
INSERT INTO PASSENGER_T VALUES ( 203,'Chitti','10/10/2016','F' );
INSERT INTO PASSENGER_T VALUES ( 204,'DumDum','10/10/2017','M' );
INSERT INTO PASSENGER_T VALUES ( 205,'Elizabeth','10/10/2011','F' );

INSERT INTO BOOK_TICKET_T VALUES (201,101,'10/10/2014',1)
INSERT INTO BOOK_TICKET_T VALUES (202,102,'10/10/2014',2)
INSERT INTO BOOK_TICKET_T VALUES (203,106,'11/03/2014',1)
INSERT INTO BOOK_TICKET_T VALUES (204,103,'11/03/2014',1)
INSERT INTO BOOK_TICKET_T VALUES (205,104,'11/03/2014',1)
INSERT INTO BOOK_TICKET_T VALUES (205,104,'01/05/2014',1)
INSERT INTO BOOK_TICKET_T VALUES (205,104,'12/02/2014',1)
```

Note: Read all questions and populate values accordingly.
c. Include constraint that DOB of passenger should be after 2010 [5]

```
ALTER TABLE PASSENGER_T ADD CONSTRAINT CHECK_PASS_DOB check (EXTRACT(YEAR FROM DOB)
> 2010)
```

c. Display the passengers who had booked the journey from Bangalore to Chennai on 03-NOV-2014. [10]

```
Select P.* from  passenger_t P,book_ticket_t t,bus_t b where t.pid = p.pid and t.journey_date =
'11/3/2014' and t.routeno = b.routeno and  b.source = 'BANGLORE' and b.destination='CHENNAI'
```

d. List the details of passengers who have traveled more than three times on the same route. [10]

```
select pid, count(routeno) from book_ticket_t group by pid having count (routeno) >= 3;
```

e. Create a view that displays the RouteNo, source, destination and journey_date which moves from Chennai to Pune. [10]

```
CREATE VIEW TRAVEL_VI AS SELECT T1.ROUTENO, SOURCE, DESTINATION, JOURNEY_DATE FROM
BUS_T T1, BOOK_TICKET_T T2 WHERE T1.ROUTENO = T2.ROUTENO and T1.SOURCE = 'CHENNAI'
and T1.DESTINATION = 'PUNE'
```

g. Create an index on PID in passenger table. [5]

```
CREATE UNIQUE INDEX PASSENGER_PID_INDEX on PASSENGER_T(PID)
```

h. Create a PL / SQL stored procedure that accepts journey_date and displays list of passengers booked ticket on that date. [20]

```
create or replace procedure find_passenger_sp
(
journeydate in date,
names out varchar
)
as
Begin
Select  S1.PNAME into NAMES from PASSENGER_T S1, BUS_T S2, BOOK_TICKET_T S3
WHERE S1.PID = S3.PID and S2.ROUTENO = S3.ROUTENO and S3.JOURNEY_DATE=journeydate;
end find_passenger_sp;
```

i. In the above created procedure, include exceptions to display "No ticket booked on specified date" for a given journey_date

```
create or replace procedure find_passenger_sp
(
journeydate in date,
names out varchar
)
as
Begin
IF '12/15/2014' = journeydate THEN
      RAISE Date_Exception;
   END IF;
Select  S1.PNAME into NAMES from PASSENGER_T S1, BUS_T S2, BOOK_TICKET_T S3
WHERE S1.PID = S3.PID and S2.ROUTENO = S3.ROUTENO and S3.JOURNEY_DATE=journeydate;
Exception
When Date_Exception Then
dbms_output.put_line('No Tickets are Booked on this date');
end find_passenger_sp;
```


Consider the following relations for a boat management application for a beach resort:
SAILOR (SID, NAME, DOB, GENDER)

```
CREATE TABLE SAILOR(  SID INT  NOT NULL, NAME VARCHAR(50), DOB DATE, GENDER VARCHAR
(1),PRIMARY KEY(SID));
```

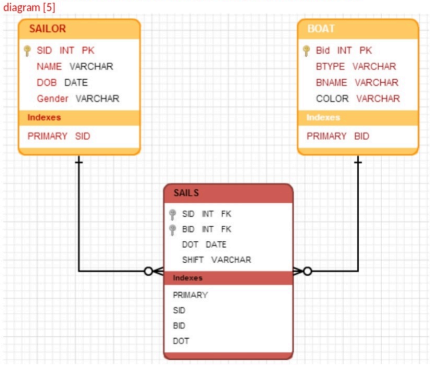
BOAT (BID, BTYPE, BNAME, COLOR)
BTYPE can take two values (D, S)
D – Deluxe
S –Super Deluxe

```
CREATE TABLE BOAT (BID INT NOT NULL, BTYPE VARCHAR (1), BNAME VARCHAR (50), COLOR
VARCHAR (20), PRIMARY KEY(BID),
CONSTRAINT CHECK_BTYPE check (BTYPE in ('B','S')))
```

SAILS (SID, BID, DOT, SHIFT)
DOT – Date of Trip
SHIFT can take two values – FN or AN
A sailor is assigned a boat on a day. A sailor is permitted to sail the boat for only one shift on a day.

```
CREATE TABLE SAILS (SID INT NOT NULL, BID INT NOT NULL, DOT DATE, SHIFT VARCHAR(2), PRIMARY
KEY(SID, BID,DOT), CONSTRAINT CHECK_SHIFT check (SHIFT in ('AN','FN')))
```

a. The primary keys are underlined. Identify the foreign keys and draw schema diagram [5]



b. Create the above mentioned tables and populate the tables [15]
Note: Read all questions and populate values accordingly.

```
INSERT INTO SAILOR VALUES (&SID,'&NAME','&DOB','&GENDER' );
INSERT INTO BOAT VALUES (&BID,'&BTYPE','&BNAME','&COLOR');
```



```
INSERT INTO SAILS VALUES (&SID,&BID,&DOT,'&SHIFT');
```

c. Include constraints for BTYPE and SHIFT as mentioned above [10]
d. Develop a SQL query to list the details of boats whose type is Super Deluxe and Color is Red. [10]

```
Select * from boat where btype = 'Super Deluxe' and Color = 'RED'
```

e. Develop a view that will keep track of sailor id, sailor name, date of trip, boat id, boat type, boat name and shift. [20]

```
Create view boat_sail_view as
Select sid, name, dot, bid, btype,bname, shift from sailor t1, boat t2, sails t3 where t1.sid = t3.sid and
t2.bid = t3.bid
```

f. Create synonym for sailor table. [5]

```
Create public synonym sailor for kgisl.sailor;
```

g. Create a PL / SQL stored function that accepts SID and returns the name of sailor [20]

```
create or replace procedure find_sname_sp
(sail_id in int, sname_out out varchar ) as
Begin
IF not exits (select SID from sailor where sid = sail_id) THEN
      RAISE Data_Exception;
   END IF;
Select  sname into sname_out from sailor where sid = sail_id;
Exception
When Data_Exception Then
dbms_output.put_line('No matching sail ids');
end find_sname_sp;
```

h. In the above created function, include exceptions to display "No such Sailor exist" when the incorrect SID is given.

```
create or replace procedure find_sname_sp
(sail_id in int, sname_out out varchar ) as
Begin
Select  sname into sname_out from sailor where sid = sail_id;
end find_sname_sp;
```

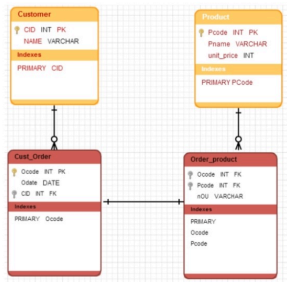
Exercise – 3

Consider the following relations for an order processing application:
CUSTOMER (CID, NAME)
PRODUCT (PCODE, PNAME, UNIT_PRICE)
CUST_ORDER (OCODE, ODATE, CID)
ORDER_PRODUCT (OCODE, PCODE, NOU)
NOU – Number of Units. An order can contain many products.
a. The primary keys are underlined. Identify the foreign keys and draw schema diagram [5]

b. Create the above mentioned tables and populate the tables [20]
Note: Read all questions and populate values accordingly.

```
Create table Customer (Cid int not null, name varchar(50), primary key(cid));
Create table Product (pid int not null, pname varchar(50), unit_price int, primary key(pid));
Create table customer_order (ocode int not null, odate date, cid int not null, primary key(ocode));
Create table order_product (ocode int not null, pcode int not null, nou varchar, primary key(ocode,
pcode));

INSERT INTO Customer VALUES (&CID,'&NAME');
INSERT INTO Product VALUES (&Pcode,'&PNAME','&unit_price');
INSERT INTO Cust_Order VALUES (&ocode,&odate,&cid);
INSERT INTO Order_Product VALUES (&ocode,&pcode,&nou);
```



c. Ensure that product names should be within Laptop, Mouse, Server, Air conditioner [5]

```
alter table product add constraint check pnname in ('Laptop', 'Mouse', 'Server', 'Air conditioner')
```

d. Develop a SQL query to list the details of products whose unit price is greater than the average price of all products [10]

```
select * from product where unit_price > avg(unit_price);
```

e. List the customer names who have orders more number of products [10]

```
select t1.name from customer t1, cust_order t2, order_product t3 where t1.cid = t2.cid and t2.ocode
= t3.ocode.
```

f. Create a view that displays the PCODE, PNAME and NOU of the product ordered [10]

```
create view product_view as
select t1.pcode, t1.pname, t2.now from product t1, Order_Product t2 where t1.pcode = t2.pcode
```

g. Create a function that accepts PCODE, Unit_Price and NOU. Calculate the total_cost of the ordered product. Return the total_cost. [20]

```
create procedure get_total_cost_sp(pcode in int, unit_price in int, nou in int, total_cost out varchar)
as
begin
Select  unit_price * nou into total_cost;
end find_sname_sp;
```

h. Create a sequence named Product_Sequence that will get incremented by 1. Use the created sequence while inserting PCODE into Product table. [20]

```
Create sequence product_sequence minvalue 1 start with 1 increment by 1 cache 20;
```

Exercise - 4

Consider the following relations for a transport management system application:
BUS (ROUTENO, SOURCE, DESTINATION)
DRIVER (DID, DNAME, DOB, GENDER)
ASSIGN_ROUTE (DID, ROUTENO, JOURNEY_DATE)

```
Create table bus (routeno int not null, source varchar(50), destination varchar(50), primary
key(routeno));
Create table driver (did int not nll, dname varchar(50), dob date, gender varchar(1), primary
key(did));
Create table assign_route(did int not null, routeno int not all, journey_date date, primary key(did,
routeno));

Insert into bus values(&routeno,'&source','&destination');
Insert into bus driver(&did,'&dname',&dob,'&gender');
Insert into bus assign_route(&did,&routeno,&journey_date);
```

a. The primary keys are underlined. Identify the foreign keys and draw schema diagram [5]
b. Create the above mentioned tables and populate the tables [15]
Note: Read all questions and populate values accordingly.
c. Include constraints that the routeNo starts with letter 'R' and gender of driver is always 'Male' [10]

```
Alter table bus add constraint check (routeno like 'R%');
Alter table driver add constraint check (gender = 'M');
```

d. Develop a SQL query to list the details of drivers who have traveled more than three times on the same route [10]

```
select t1.did, dname,dod, gener from driver t1, assign_route t2 where t1.did = t2.did group by t2.did
having count (routeno) >= 3;
```

e. Create a sequence named Driver_Sequence that will get incremented by 1. Use the created sequence while inserting DID into Driver table. [20]

```
Create sequence driver_sequence minvalue 1 start with 1 increment by 1 cache 20
```

f. Create a view that displays the DID, DNAME assigned for RouteNo 'R5' on 02-NOV-2014 [20]

```
create view driver__view as
select t1.did, t1.dname, t2.routeno from driver t1, assign_route t2 where t1.did = t2.did and
t2.routeno = 'R5' and t2. Journeydate ='02-nov-2014';
```


g. Create a procedure that displays the details of all drivers. [20]

```
create procedure get_drivers_sp as
begin
select * from drivers;
end get_drivers_sp;
```

Exercise – 5

5. Consider the following relations for a transport management system application:
DRIVER (DCODE, DNAME, DOB, GENDER) CITY (CCODE, CNAME)
TRUCK (TRUCKCODE, TTYPE)
TTYPE can take two values ('L','H')
L-Light
H- Heavy
Each truck is assigned a unique truck code. There can be many trucks belonging to the same truck type.
DRIVE_TRUCK (TRUCKCODE, DCODE, DOT, CCODE)
DOT – Date of Trip

a. The primary keys are underlined. Identify the foreign keys and draw schema diagram [5]

b. Create the above mentioned tables and populate the tables [20]

Note: Read all questions and populate values accordingly.

```
Create table driver (dcode int not null, dname varchar (50), dob date, gender varchar(10), primary
key(dcode));

Create table city (ccode int not null, cname varchar(50), primary key(ccode));

Create table truck (truckcode int not null, ttype varchar(50), primary key(truckcode));

Create table drive_truck (truckcode int not null, dcode int not null, dot date, ccode int not null,
primary key (truckcode,dcode, dot));
```

c. Include the constraint as mentioned above and the gender of driver is always 'male'. [10]

```
Alter table driver add constraint check (gender = 'male')
```

d. Develop a SQL query to list the details of each driver and the number of trips traveled. [10]

```
select  dname, count(dot) from driver t1, drivetruck t2 where t1.dcode = t2.dcode group by
t2.dcode
```

e. Create an index on truck_code in Drive_truck table [5]

```
Create unique index trk_cde on Driver_Truck(truck_code);
```

f. Create a view that displays the Driver details and also the city in which he drives a truck [20]

```
Create view driverDetailsView as
Select t1.dname, t2.city from driver t1, city t2, driver_truck t3 where t1.dcode=t3.dcode and
t3.code = t3.ccode;
```


g. Create a procedure that displays the details of all drivers, the truck_code and DOT. Use cursors appropriately. [30]

```
  create procedure display_employee as
Begin
  DECLARE  CURSOR employee_cur
  IS
    SELECT t1.dname, t1.dob, t1.gender, t2.truckcode,t2.dot FROM driver t1, drive_truck t2 where
t1.dcode = t2.dcode;

  l_name   employee_cur%ROWTYPE;
  l_dob employee_cur%ROWTYPE;
  l_gender employee_cur%ROWTYPE;
  l_truckcode employee_cur%ROWTYPE;
  l_dot employee_cur%ROWTYPE;
BEGIN
  OPEN employee_cur;
  LOOP
    FETCH employee_cur INTO l_name, l_dob,l_gender,l_truckcode, l_dot;
    EXIT WHEN employee_cur%NOTFOUND;
dbms_output.put_line( l_name, l_dob,l_gender,l_truckcode, l_dot);
  END LOOP;
  CLOSE employee_cur;
END;
End;
```

Exercise – 6

Consider the following relations for an order-processing database application in a company:

```
CUSTOMER (CUSTOMERNO  VARCHAR2 (5), CNAME VARCHAR2 (30), CITY VARCHAR2 (30))
Implement a check constraint to check CUSTOMERNO starts with 'C'
CUST_ORDER  (ORDERNO  VARCHAR2  (5),  ODATE  DATE,  CUSTOMERNO REFERENCES CUSTOMER,
ORD_AMT NUMBER (8))
Implement a check constraint to check ORDERNO starts with 'O'
ITEM (ITEMNO  VARCHAR2  (5),  ITEM_NAME  VARCHAR2  (30), UNIT_PRICE NUMBER (5))
Implement a check constraint to check ITEMNO starts with 'I'
ORDER_ITEM  (ORDERNO  REFERENCES  CUST_ORDER,  ITEMNO REFERENCES ITEM, QTY
NUMBER (3))
SHIPMENT (ORDERNO  REFERENCES CUST_ORDER, ITEMNO  REFERENCES ITEM, SHIP_DATE DATE)
Here, ORD_AMT refers to total amount of an order (ORD_AMT is a derived attribute);
ODATE is the date the order was placed; SHIP_DATE is the date an order is shipped.
```

a. The primary keys are underlined. Identify the foreign keys and draw schema diagram [5]

b. Create the above mentioned tables and populate the tables [30]

Note: Read all questions and populate values accordingly.


```
Create Table Customer (Customerno  Varchar2 (5), Cname Varchar2 (30), City Varchar2 (30), Primary
Key(Customerno));

Create Table Cust_Order  (Orderno  Varchar2  (5),  Odate  Date,  Customerno Varchar2 (5), Ord_Amt
Number (8))

Create Table Item (Itemno  Varchar2  (5),  Item_Name  Varchar2  (30), Unit_Price Number (5));
Create Table Order_Item      (Orderno  Varchar2  (5),  Itemno Varchar2 (5), Qty Number (3))
Create Table Shipment (Orderno  Varchar2(5), Itemno  Varhar2(5), Ship_Date Date)

Insert into Customer Values ('&Customerno','&Cname','&City');
Insert into Cust_Order ('&Orderno',&Odate,'&Customerno',&Ord_Amt);
Insert into Item values ('&Itemno',&Item_Name',&UnitPrice);
Insert into Shipment ('&OrderNo','&ItemNo',&ShipDate);
```

c. Include the constraint as mentioned above. [10]

```
Alter table Customer add constraint check (CustomerNo like 'C%');
Alter table Order add constraint check (OrderNo like 'O%');
```

d. Develop a SQL query to list the order number and number of items in each order [10]

```
Select t1.OrderNo, t2.Qty from Cust_Order t1, Order_Item t2 where t1. Orderno = t2.orderno;
```

e. Create a synonym on for CUST_ORDER table [5]

```
Create Synonym Cust_Order for Kgisl.Cust_Order;
```

f. Create a view that will keep track of the details of each customer and the number of orders placed by each customer [20]

```
Create view customerOrderView as
Select t1.customerNo, t1.cname, t2.orderno from customer t1, cust_order t2 where t1.customerNo =
t2.CustomerNo;
```

g. Develop a database trigger that will not permit to insert more than six records in the CUST_ORDER relation for a particular order. (An order can contain a maximum of six items). [20]

```
Create trigger cust_order_insert after insert on cust_order
for each row
declare count_number;
begin
select count(orderno) into count_number from customer t1, customer_order t2 where
    t1.customerno = t2.customerno;
   IF (count_number >6) THEN
      DBMS_OUTPUT.PUT_LINE('Max order limit reached for the customer.');
rollback;
   END IF;
end;
```

Exercise – 7

7. Consider the following relational schema for a banking database application:

```
CUSTOMER (CID, CNAME)

ACCOUNT (ANO, ATYPE, BALANCE, CID)
```

An account can be a savings account or a current account. Check ATYPE in 'S' or 'C'. A customer can have both types of accounts.

```
TRANSACTION (TID, ANO, TTYPE, TDATE, TAMOUNT)

TTYPE CAN BE 'D' OR 'W'
```

D- Deposit; W – Withdrawal

a. The primary keys are underlined. Identify the foreign keys and draw schema diagram [5]

b. Create the above mentioned tables and populate the tables [15]

```
Create table Customer (Cid Number, Cname Varchar2(20), Primary key(Cid));

Create table Account (Ano Number, Atype Varchar2(20), Balance Number, CID Number, Primary
key(Ano));

Create table Transaction (TID Number, Ano Number, Ttype varchar2(20), Tdate date, Tamount
number, primary key(tid));


Insert into Customer Values (&Cid,'&CName');

Insert into Account Values(&Ano,'&Atype',&Balance,%Cid);

Insert into Transaction Values (&Tid, &Ano,'&Ttype',&Tdate,&Tamount);
```

Note: Read all questions and populate values accordingly.

c. Include the constraints as mentioned above. [10]

```
Alter table Account add constraint check (Atype in ('S','C','A'));

Alter table Transaction add constraint check (Tid in ('D','W'));
```

d. Write a query that lists the customer details and the number of accounts each customer has. [10]

```
Select t1.cid,t1.cname, count(t2.Ano) from Customer t1, Account t2 where t1.Cid = T2 .Cid group by
t2.cid
```

e. Create a sequence named Customer_Sequence which gets incremented by 10 and use this sequence to give values of CID in customer table. [10]

```
Create sequence Customer_Sequence minvalue 1 start with 1 increment by 1 cache 20;
```

f. Create a view that will keep track of the details of each customer and account details who have both savings and current account. [10]

```
Create view CustomerAccount_View as

Select t1.* , t2.* from Customer t1, Account t2 where t1.cid = t2.cid and t2.Atype in ('S','C');
```


g. Develop a database procedure that will accept transaction id, account number, transaction type, transaction date and transaction amount as input and insert a record into

TRANSACTION table subject to the following conditions:

i. If TTYPE ='D' the value of BALANCE in the ACCOUNT table must be incremented by the value of TAMOUNT

ii. If TTYPE ='W' the value of BALANCE in the ACCOUNT table must be decremented by the value of TAMOUNT.

If a minimum balance of Rs. 2000/- will be maintained for a savings account and a minimum balance of Rs. 5000/- will be maintained for a current account else appropriate messages must be displayed [30]

i. In the above created procedure, if TTYPE = 'W', and transaction amount is > available balance, raise exceptions to display "Amount > available Balance" [10]

Exercise – 8

Consider the following relational schema for a banking database application:

```
CUSTOMER (CID, CNAME)

BRANCH (BCODE, BNAME)

ACCOUNT (ANO, ATYPE, BALANCE, CID, BCODE)
```

An account can be a savings account or a current account. Check ATYPE in 'S' or 'C'. A customer can have both types of accounts.

```
TRANSACTION (TID, ANO, TTYPE, TDATE, TAMOUNT)

TTYPE CAN BE 'D' OR 'W'
```

D- Deposit; W – Withdrawal

a. The primary keys are underlined. Identify the foreign keys and draw schema diagram [5]

b. Create the above mentioned tables and populate the tables [20]

Note: Read all questions and populate values accordingly.

```
Create table Customer (CID Number, Cname Varchar2(20), primary key(Cid));

Create table Branch (Bcode Number, Bname Varchar2(20), Primary key(Bcode));

Create Table Account (Ano Number, Atype Varchar2(20), Balance Number, CID Number, BCode
Varchar2(20), Primary key(Ano));

Create Table Transaction (Tid Number, Ano Number, Ttype Varchar2(20), Tdate date, Tamount
Number, Primary Key(Tid));

Insert into Customer Values (&Cid, '&Cname');

Insert into Branch Values (&Bcode, '&Bname');
```

```
Insert into Account Values (&Ano, '&Atype' , &Balance, &Cid, '&Bcode');

Insert into Transaction Values (&Tid, &Tno, '&TType', &tdate,&tamount);
```

c. Include the constraints as mentioned above. [10]

```
Alter table Account add constraint Acc_cnstr Check (Atype in ('S','C'));

Alter table Transaction add constraint Trans_cntr check (Ttype in ('D','W'));
```

d. Develop a SQL query to list the details of branches and the number of accounts in each branch. [10]

```
Select t1.Bcoe, t1.Bname, count(t2.Ano) from Branch t1, Account T2 where t1.Bcode = t2. Bcode group by t2.Ano;
```

e. Develop a SQL query to list the details of customers who have performed three transactions on a day [15]

```
Select t1.Cname, count(t3.Ano) from Customer t1, Account t2, Transaction t3 where t1.cid = t2. cid and t2.ano = t3. Ano group by t3. Ano, t3.tdate having count(t3.Ano) > 3;
```

f. Create a view that will keep track of the details of each customer and account details who have both savings and current account. [10]

```
Create View customerAccount_View as

Select t1.Cid, t1.Cname from Customer t1, Account t2 where t1.cid = t2.cid and t1.cid in (select cid from Account where Atype = 'S' and Cid in (Select Cid from Account where Atype = 'C'));
```

g. Develop a database trigger that will update the value of BALANCE in ACCOUNT table when a record is inserted in the transaction table. Consider the following cases:

i. If TTYPE ='D' the value of BALANCE in the ACCOUNT table must be incremented by the value of TAMOUNT

ii. If TTYPE ='W' the value of BALANCE in the ACCOUNT table must be decremented by the value of TAMOUNT.

If a minimum balance of Rs. 2000/- will be maintained for a savings account and a minimum balance of Rs. 5000/- will be maintained for a current account else appropriate messages must be displayed [30]

```
create trigger Account_insert
after insert
on Account
for each row
begin
   IF (Ttype  = 'D') THEN
            update Account set Balance = Balance+Tamount where Ano = ano;
   ELSIF (Ttype  = 'W') THEN
            update Account set Balance = Balance-Tamount where Ano = ano;
   END IF;
```



```
   IF (Atype = 'S' and Balance < 2000) Then
         dbms_output.put_line('Savings Account Balance is less the required minium account ')
   ELSIF (Atype = 'C' and Balance < 5000) Then
         dbms_output.put_line('Current Account Balance is less the required minium account ')
   END IF;
end;
```

Exercise -9

Consider the following relational schema for a library management system:

BOOK (BOOKID, TITLE, PUBLISHERCODE, NO_OF_COPIES)

PUBLISHER (PUBLISHERCODE, PUBLISHER_NAME)

AUTHOR (AUTHORID, AUTHOR_NAME)

BOOK_AUTHOR (BOOKID, AUTHORID)

BORROWWER (CARDNO, NAME)

BOOK_LOAN (BOOK_ID, CARDNO, DATEOUT, DUEDATE, STATUS)

Implement a Check Constraint for STATUS ('R' – Returned, 'T' – To be returned)

a. The primary keys are underlined. Identify the foreign keys and draw schema diagram [5]

b. Create the above mentioned tables and populate the tables [25]

Note: Read all questions and populate values accordingly.

```
Create Table Book (Bookid Number, Title Varchar2(20), publisherCode Number, No_of_Copies Number, Primary Key(Bookid));

Create Publisher (PublisherCode Number, PublisherName Varchar2(20), Primary Key(PublisherCode));

Create Table Author (authorId Number, Authur_Name Varchar2(20), Primary Key(Authorid));

Create Table Book_Author (Bookid Number, AuthorId Number, Primary key (Bookid));

Create Table Borrower (CardNo Number, Name Varchar2(20), Primary Key(CardNo));

Create Table Book_Loan (Book_ID Number, CardNo Number, DateOut Date, DueDate Date, Status Varchar2(20), Primary Key(Book_id, Cardno, Dateout));

Insert into Book Values(&Bookid, '&Title',&PublisherCode, &No_of_Copies);
```

c. Include the constraints as mentioned above. [5]

```
Alter Table Book_Loan Constraint Bk_status  Check (Status in ('R','T'));
```

d. Develop a SQL query to list the details of borrowers who do not have any books checked out. [5]


```
Select t1. Name from Borrower t1, Book_Loan t2 where t1.cardNo = t2.CardNo and t2.Status ='R';
```

e. Develop a SQL query to list the details of borrowers who have more than five books checked out. [10]

```
Select t1. Name, count(t2.status) from Borrower t1, Book_Loan t2 where t1.cardNo = t2.CardNo and t2.Status ='T' group by t2.cardNo having count(cardNo) > 5;
```

f. Create an index on BookID in Book_Loan table [10]

```
Create unique index BookidIndx on Book_Loan(Bookid);
```

g. Create a view that will keep track of the card number, card holders name and number of books borrowed (Number of books with status 'T') [10]

```
Create View BorrowerView as

Select t1. Name, count(t2.status) from Borrower t1, Book_Loan t2 where t1.cardNo = t2.CardNo and t2.Status ='T' group by t2.cardNo ;
```

h. Create a procedure named Author_Details that accepts the BookID and displays the author ID, author name and also the status of the book. [30]

```
create or replace procedure Author_details_sp ( bookid in Number, Authorid out varchar2(20), Name out varchar2(20),Status out Varchar2(20) )

as Begin

Select t1. Authorid, t2.Name, t3.status from Author t1, Author_book t2, Book_Loan t3 where t1.authorid = t2.Authorid and t1.bookid = Bookid and t3.Bookid = t2.Bookid;

end find_passenger_sp;
```

Exercise – 10

Consider the following Staff relational schema:

STAFF (STAFFNO, NAME, DOB, GENDER, DOJ, DESIGNATION,

BASIC_PAY, DEPTNO)

GENDER must take the Value 'M' or 'F'

DEPT (DEPTNO, NAME)

SKILL (SKILL_CODE, DESCRIPTION, CHARGE_OUTRATE)

STAFF_SKILL (STAFFNO , SKILL_CODE)

PROJECT (PROJECTNO, PNAME, START_DATE, END_DATE, BUDGET, PROJECT_MANAGER_STAFFNO)

WORKS (STAFFNO, PROJECTNO, DATE_WORKED_ON, IN_TIME,

OUT_TIME)

a. The primary keys are underlined. Identify the foreign keys and draw schema diagram [5]

b. Create the above mentioned tables and populate the tables [30]

Note: Read all questions and populate values accordingly.

```
Create table Staff (StaffNo Number, Name Varchar2(20), DOB date, Gender Varchar2(20), DOJ Date, Designation Varchar2(20), Primary Key(StaffNo));

Create Table Dept (DeptNo Number, Name Varchar2(20), Primary Key(DeptNo));

Create Table Staff_Skill(Skill_code Numberr, Description Varchar2(20), Primary Key(Skill_Code));

Create Table Staff_Skill (StaffNo Number, Skill_code Number, Primary Key(StaffNo, Skill_Code));
```

c. Include the constraints as mentioned above. [5]

```
Alter Table Staff add Constratint GenderChk Check(Gender in ('M','F'));
```

d. Develop a SQL query to list the details of staff who earn less than the basic pay of all staff. [10]

```
Select t1. Name from Staff t1 where t1.Basic_pay < Min(Basic_Pay);
```

e. Create a view that keeps track of DeptNo, DeptName and number of staff in each department. [10]

```
Create View staffDeptView as

Select t1.DeptNo, t1.DeptName, Count(t2.StaffNo) from Dept t1, Staff t2 where t1.staffno = t2. Staff no and t1.DeptNo = t2.DeptNo group by t1.deptNo;
```

f. Develop a SQL query to list the details of staff who have more than three skills. [5]

```
Select t1.StaffNo, t1.Name, t1.Designation from Staff t1, StaffSkill t2, Skill t3 where t1.Staffno= t2.Staffno and t2.Skillcode =t3.Skillcode and count (t3.skillcode ) > 3 group by t3.skillcode;
```

g. Create an index on StaffNo in Works table [5]

```
Create unique index StaffNoIndx on Works(StaffNo);
```

h. Develop a procedure Staff_Increment that will accept staff number and increment amount as input and update the basic pay of the staff in the staff table. [20]

i. In the above procedure include exception in the procedure that will display a message "Staff has basic pay null" if the basic pay of the staff is null and display a message "No such staff number" if the staff number does not exist in the staff table. [10]

```
Create procedure staff_increment (Staffno in Number,IncrementAmount in Number )
as begin
   update Table Staff set BasicPay = BasicPay+IncrementAmount where StaffNo = staffno;
```



```
   IF (BasicPay = Null) Then
         dbms_output.put_line('Baic pay of staff is Null. ')
   ELSIF (if not exist(Select StaffNo from Staff where StaffNo = StaffNo) ) Then
         dbms_output.put_line('No Such Staff Number Exist ')
   END IF;
End ;
```

Exercise – 11

Consider the following relational schema for a company database application:
EMPLOYEE (ENO, NAME, GENDER, DOB, DOJ, DESIGNATION, BASIC, DEPT_NO, PAN, SENO)
Implement a Check Constraint for GENDER
PAN – Permanent account Number
SENO – Supervisor Employee Number
DEPARTMENT (DEPT_NO, NAME, MENO)
MENO - Manager Employee Number
PROJECT (PROJ_NO, NAME, DEPT_NO)
WORKSFOR (ENO, PROJ_NO, DATE_WORKED, HOURS)
a. The primary keys are underlined. Identify the foreign keys and draw schema diagram [5]
b. Create the above mentioned tables and populate the tables [20]
Note: Read all questions and populate values accordingly.

```
Create Table Employee (Eno Number, Gender Varchar2(10), DOB Date, DOJ Date, Designation Varchar2(20), Basic Number, Dept_no Number, Primary Key(Eno));
Create Table Department (DeptNo Number, Name Varchar2(20), Memo Varchar2(20), Primary Key(DeptNo));
Create Table Project (ProjectNo Number, Name Varchar2(20), DeptNo Number, Primary Key(ProjNo));
```

c. Include the constraints as mentioned above. [5]

```
Alter Table Employee Add Constraint GenderChk Check (Gender in ('M','F'));
```

d. Develop a SQL query to list the details of department which has more than 3 employees working for it. [10]

```
Select t1.DeptNo, T1.DeptName where Department T1, Employee T2 where t1.DeptNo = T2.DeptNo and group by t.1DeptNo having count(t1.deptno) >3;
```

e. Create a view that keeps track of DeptNo, DeptName and number of employees in each department. [10]

```
Create View deptView as
Select t1.DeptNo, T1.DeptName, count(t2.Eno) from Department T1, Employee T2 where t1.DeptNo = T2.DeptNol
```

f. Develop an SQL query to list the departments and the details of manager in each department. [5]

```
Select t1.Name, t2.Designation from Departmnet t1, Employee t2 where t1.DeptNo = T2.DeptNo and T2.Designation = 'Employee';
```

g. Create an index on EmpNo in WorksFor table [5]

```
Create unique index EmpnoIndx on WorksFor(EmpNo);
```


h. Develop a procedure ***Employee_Increment*** that will accept Employee number and increment amount as input and update the basic pay of the employee in the employee table. [20]
i. In the above procedure include exception in the procedure that will display a message "Employee has basic pay null" if the basic pay of the employee is null and display a message "No such Employee number" if the employee number does not exist in the employee table. [10]
j. Create a database trigger that will not permit to insert values into Employee table if DOJ is less than DOB. [10]

```
Create procedure staff_increment (empNo in Number,IncrementAmount in Number )
as begin
   update Table Employee set BasicPay = BasicPay+IncrementAmount where EmpNo = empNo;

   IF (BasicPay = Null) Then
         dbms_output.put_line('Baic pay of staff is Null. ')
   ELSIF (if not exist(Select EmpNo from Employee where EmpNo = empNo) ) Then
         dbms_output.put_line('No Such Employee Number Exist ')
   END IF;
End ;
```

Exercise – 12

Consider the following relational schema for a Product Sales database application:

Product (Prodid, Prodesc, Price, Stock)

Purchase (Purid, Proid, qty, supplierName)

Sales (Saleid, Proid, qty, custname)

a. The primary keys are underlined. Identify the foreign keys and draw schema diagram [5]

b. Create the above mentioned tables and populate the tables [20]

Note: Read all questions and populate values accordingly.

```
Create Table Product (ProductID Number. Prodesc Varchar2(20), Price Number, Stock Number, Primary Key(Prodid));

Create Table Purchase (Purid Number, Proid Number, Qty Number, SupplierName Varchar2(20), Primary Key(Purid));

Create Table Sales (Salesid Number, Proid Number, Qty Number, CustName Varchar2(20));

Insert into Product Values(&Prodid, '&Prodesc',&Price,&Stock);

Insert into Purchase Values(&Purid,&Proid,&qty,'&SupplierName');

Insert into Sales (&Saleid, &PRoid, &qty, '&CustName');
```

c. Include the constraint on Saleid that it starts with letter 'S'. [5]

```
Alter Table Sales add Constraint Saleidx Check (Saleid like 'S%');
```

d. Display the ProdID and the sum of quantity purchased for each product. [10]

Select t1.Prodid, prodesc SUM(Qty) product t1, Purchase t2 where t1.proid =p2.proid;

e. Create a view that keeps track of Prodid, price, Purid, qty and customerName who made the purchase. [20]

Create view productView as

Select t1.Proid, t1.price, t2.purid, t2.qty, t2.custName from Product t1, sales t2 where t1.prodid = t2.prodid;

f. Create a sequence named Product_Sequence that gets incremented by 10 and use it for inserting Prodid values in Product table. [10]

Create sequence product_sequence minvalue 10 start with 10 increment by 10 cache 20;

g. Develop a procedure named Product_Sales that accepts a prodid and displays all the sales and purchase records of it. [20]

h. In the above procedure include exception in the procedure that will display a message "No such Product ID" if the given product id does not exist in the product table. [10]

```
Create procedure product_sales (pid in Number )
as begin
  Select t1.Proid, t1.price, t2.purid, t2.qty, t2.custName from Product t1, sales t2 where t1.prodid =
t2.prodid; and t1.prodid = pid;

   IF (if not exist(Select Prodid from Product where Prodid = Pid) ) Then
          dbms_output.put_line('No Such Product Exist ')
   END IF;
End ;
```

Exercise – 13

Consider the following relational schema for a Product Sales database application:

Product (Prodid, Prodesc, Price, Stock)

Purchase (Purid, Proid, qty, supplierName)

Sales (Saleid, Proid, qty, custname)

a. The primary keys are underlined. Identify the foreign keys and draw schema diagram [5]

b. Create the above mentioned tables and populate the tables [20]

Note: Read all questions and populate values accordingly.

Create Table Product (ProductID Number. Prodesc Varchar2(20), Price Number, Stock Number, Primary Key(Prodid));

Create Table Purchase (Purid Number, Proid Number, Qty Number, SupplierName Varchar2(20), Primary Key(Purid));

Create Table Sales (Salesid Number, Proid Number, Qty Number, CustName Varchar2(20));


```
Insert into Product Values(&Prodid, '&Prodesc',&Price,&Stock);

Insert into Purchase Values(&Purid,&Proid,&qty,'&SupplierName');

Insert into Sales (&Saleid, &PRoid, &qty, '&CustName');
```

c. Include the constraint on Saleid that it starts with letter 'S'. [5]

```
Alter Table Sales add Constraint Saleidx Check (Saleid like 'S%');
```

d. Display the ProdID and the sum of quantity purchased for each product. [10]

```
Select t1.Prodid, prodesc SUM(Qty) product t1, Purchase t2 where t1.proid =p2.proid;
```

e. Create a view that keeps track of Prodid, price, Purid, qty and customerName who made the purchase. [20]

```
Create view productView as

Select t1.Proid, t1.price, t2.purid, t2.qty, t2.custName from Product t1, sales t2 where t1.prodid =
t2.prodid;
```

f. Create a sequence named Product_Sequence that gets incremented by 10 and use it for inserting Prodid values in Product table. [10]

```
Create sequence product_sequence minvalue 10 start with 10 increment by 10 cache 20;
```

g. Develop a procedure named Product_Sales that accepts a prodid and displays all the sales and purchase records of it. [20]

"No such Product ID" if the given product id does not exist in the product table. [10]

```
Create procedure product_sales (pid in Number )
as begin
  Select t1.Proid, t1.price, t2.purid, t2.qty, t2.custName from Product t1, sales t2 where t1.prodid =
t2.prodid; and t1.prodid = pid;

   IF (if not exist(Select Prodid from Product where Prodid = Pid) ) Then
          dbms_output.put_line('No Such Product Exist ')
   END IF;
End;
```

Exercise - 14

Consider the following relational schema for a Loan database application:
Customer (Custid, Custname, Age, phno)
Loan (Loanid, Amount, Custid)
a. The primary keys are underlined. Identify the foreign keys and draw schema diagram [5]
b. Create the above mentioned tables and populate the tables [10]
Note: Read all questions and populate values accordingly.

```
Create Table Customer (Custid Number, CustName Varchar2(20), Age Number, Phno Number,
Primary Key(Custid));
Create Table Loan(Loanid Number, Amount Number, Custid Number, Primary Key (Loanid));
```


c. Include the constraint on Loanid that it starts with letter 'L'. [5]

```
Alter Table Loan add constraint loanidx Check (Loanid like 'L%');
```

d. Display the list of the customerids and total Loan amount taken [10]

```
Select t1.Custid, Sum(t2.Amount) from Customer t1, Loan t2 where t1. Custid = T2.Custid group
by t2.custid;
```

e. Display the CustId and CustName who have taken less than 2 loans [10]

```
Select t1.Custid, t1.CustName from Customer t1, Loan t2 where t1. Custid = T2.Custid having
count(t2.Custid <2);
```

f. Create a view that keeps track of Custid, Custname, loanid and loan amount. [20]

```
Create View CustView as
Select t1.Custid, t1.CustName,t2.Loanid,t2.Amount from Customer t1, Loan t2 where t1. Custid
= T2.Custid
```

g. Create a sequence named Customer_Sequence that gets incremented by 3 and use it for inserting Custid values in Customer table. [10]

```
Create sequence Customer_sequence minvalue 1 start with 3 increment by 3 cache 20;
```

h. Develop a function named ***Customer_Loan*** which accepts Loanid as input and displays Custid, CustName and loan_amount. [20]

```
Create procedure CustLoan_sp (Lnid in Number )
as begin
  Select t1.Custid, t1.CustName, t2.amount from Customer t1, Loan t2 where t1.Custid =
t2.Custid; and t2.loanid = Lnid
End;
```

Exercise – 15

Consider the following relational schema for a Loan database application:

Customer (Custid, Custname, Age, phno)

HLoan (HLoanid, Amount, Custid)

VLoan (VLoanid, Amount, Custid)

Where HLoan is Housing loan and VLoan is a Vechile loan.

a. The primary keys are underlined. Identify the foreign keys and draw schema diagram [5]

b. Create the above mentioned tables and populate the tables [10]

Note: Read all questions and populate values accordingly.

```
Create Table Customer (Custid Number, Custname Varchar2(20), Age Number, phno Number, Primary
Key (Custid));

Create table HLoan (HLoanid Number , Amount Number, Custid Number, Primay key(HLoanid));

Create table VLoan (VLoanid Number , Amount Number, Custid Number, Primay key(VLoanid));
```


c. Include the constraint on HLoanid that it starts with letter 'H' and VLoanid starts with letter 'V'. [5]

```
Alter table HLoan add constraint HInidx Check (HLoanid like ('H%));

Alter table VLoan add constraint VInidx Check (VLoanid like ('V%));
```

d. Display the number of VLoan taken by a particular customer id [10]

```
Select VLoanid, amount from Vloan where Customer id = V234;
```

e. Display the list of the customerids and total HLoan amount taken. [10]

```
Select HLoanid, amount from Hloan;
```

f. Create a view that keeps track of customer details who have taken both HLoan and VLoan. [20]

```
Create View customerLoan as

Select t1.CustName from Customer t1, HLoan t2, VLoan t2 where t1.custid= t2.custid and t2.custid =
t3.custid;
```

g. Create a sequence named Customer_Sequence that gets incremented by 3 and use it for inserting Custid values in Customer table. [10]

```
Create sequence Customer_sequence minvalue 1 start with 3 increment by 3 cache 20;
```

h. Develop a procedure named Customer_Loan which accepts HLoanid as input and displays Custid, CustName and loan_amount of HLoan. [20]

i. In the above procedure include exceptions to display "No such HLoanid" when incorrect Hloanid is given. [10]

```
Create procedure CustLoan_sp (HLnid in Number )
as begin
  Select t1.Custid, t1.CustName, t2.loan_amount from Customer t1, HLoan t2 where t1.Custid =
t2.Custid; and t2.loanid = HLnid;

   IF (if not exist(Select HLoanid from HLoan where HLoanid = HLnid) ) Then
          dbms_output.put_line('No Such HLoanid Exist ')
   END IF;
End ;
```

Exercise – 16

Consider the following relational schema for a Loan database application:

Customer (Custid, Custname, Addr, phno,pan_no)

Loan (Loanid, Amount, Interest,Custid)

Account (Accid, Accbal, Custid)

a. The primary keys are underlined. Identify the foreign keys and draw schema diagram [5]

b. Create the above mentioned tables and populate the tables [10]

```
Create Table Customer (Custid Number , Custname Varchar2(20), Addr Varchar2(20), phno
Number ,pan_no Varchar2(20), Primary Key(Custid));

Create Table Loan (Loanid Number , Amount Number , Interest Number, Custid Number, Primary Key
(Loanid));

Create Table Account (Accid Number, Accbal Varchar2(20), Custid Number , Primary Key(Accid));
```

Note: Read all questions and populate values accordingly.

c. Include the constraint on Custid that it starts with letter 'C' [5]

```
Alter Table Customer add constraint custidctrn Check (Custid like 'C%');
```

d. Display the customer id, name and account balance. Sort the output using custid [10]

```
Select t1.Custid, t1.CustName, t2.Amount from Customer t1, Loan t2 where t1.Custid = T2.custid;
```

e. Display the accounts of custids 'C01','C02','C03' [10]

```
Select * from Account where Accid in ('C01','C02','C03');
```

f. Display the custid who has account balance larger than other customers [5]

```
Select t1.Custid from Customer t1, Account t1 where t1.Custid  = t2.custid and t2.Accbal =
max(Accbal);
```

g. Create an index on Accid of Account table. [5]

```
Create unique index Accid_indx on Account (Accid);
```

h. Create a view that keeps track of customer id, loan amount and account balance. [20]

```
Create View CustomerView as

Select t1.Custid, t1.CustName, t2.Amount from Customer t1, Loan t2 where t1.Custid = T2.custid;
```

i. Develop a procedure named Customer_Loan that displays all the loan details [20]

j. In the above procedure include exceptions to display "No such Loanid" when incorrect loanid is given. [10]

```
Create procedure CustLoan_sp (Lnid in Number )
as begin
  Select t1.Custid, t1.CustName, t2.loan_amount from Customer t1, Loan t2 where t1.Custid =
t2.Custid; and t2.loanid = Lnid;

   IF (if not exist(Select Loanid from Loan where Loanid = Lnid) ) Then
          dbms_output.put_line('No Such Loanid Exist ')
   END IF;
End ;
```


Exercise – 17

Consider the following relational schema for a Sales database application:

Product (Prodid, Prodesc, Price, Stock)

Purchase (Purid, Proid, qty, supplierName)

Sales (Saleid, Proid, qty, custname)

a. The primary keys are underlined. Identify the foreign keys and draw schema diagram [5]

b. Create the above mentioned tables and populate the tables [20]

Note: Read all questions and populate values accordingly.

```
Create Table Product (ProductID Number. Prodesc Varchar2(20), Price Number, Stock Number,
Primary Key(Prodid));

Create Table Purchase (Purid Number, Proid Number, Qty Number, SupplierName Varchar2(20),
Primary Key(Purid));

Create Table Sales (Salesid Number, Proid Number, Qty Number, CustName Varchar2(20));

Insert into Product Values(&Prodid, '&Prodesc',&Price,&Stock);

Insert into Purchase Values(&Purid,&Proid,&qty,'&SupplierName');

Insert into Sales (&Saleid, &PRoid, &qty, '&CustName');
```

c. Include the constraint on Saleid that it starts with letter 'S'. [5]

```
Alter Table Sales add Constraint Saleidx Check (Saleid like 'S%');
```

d. Display the names who are both supplier as well as customer [10]

```
Select t1.CustName from Sales t1, Purchase t2 where t1.Custname = t2.Supplier Name;
```

e. Display the amount (price * qty) of Products in each Sales. [10]

```
Select t1.price * t2. Qty from Product t1, Purchase t2, Sales t3 where t1. Proid = t2.proid and t2.proid
= t3.proid group by t3.proid
```

f. Create a view which displays Product ids and sum of quantity in sales [20]

```
Create View productSales as
Select t1.prodid, SUM( t3. Qty) from Product t1, Purchase t2, Sales t3 where t1. Proid = t2.proid and
t2.proid = t3.proid group by t3.proid
```

g. Create a sequence named Product_Sequence that gets incremented by 10 and use it for inserting Prodid values in Product table. [10]

```
Create sequence product_sequence minvalue 1 start with 10 increment by 10 cache 20;
```

Exercise – 18

Consider the following relational schema for a Loan database application:

Customer (Custid, Custname, Age, phno)

Loan (Loanid, Amount, Custid, Emi)

a. The primary keys are underlined. Identify the foreign keys and draw schema diagram [5]

b. Create the above mentioned tables and populate the tables [10]

Note: Read all questions and populate values accordingly.

```
Create Table Customer (Custid Number, Custname Varchar2(20), Age Number, phno Number, Primary
Key(Custid));

Create Table Loan (Loanid Number, Amount Number, Custid Number, EMI Number, Primary
Key(Loanid));
```

c. Include the constraint on Custid that it starts with letter 'C'. [5]

```
Alter table Customer Add constraint CustCnstrn Check (Custid like 'C%');
```

d. Update the loan amount by increase in 2 % for all customers [10]

```
Update table Loan set Amount = Amount*00.2;
```

e. Display the custid and Custname whose loan amount lies in the range of 30,000 to 50,000

```
Select t1.Custid,t1.CustName, T2.Amount from Customer t1, Loan t2 where t1.Custid = t2.custid and
t2. Amount between 30000 and 50000;
```

f. Display the CustId and CustName who have taken less than 2 loans [10]

```
Select t1.Custid,t1.CustName, T2.Amount from Customer t1, Loan t2 where t1.Custid = t2.custid and
count(t2.custid) <2 group by t2.Custid;
```

g. Create a view that keeps track of Custid, Custname, loanid and loan amount. [20]

```
Create View CustomerView as
Select t1.Custid,t1.CustName, T2.Amount from Customer t1, Loan t2 where t1.Custid = t2.custid
```

h. Create a sequence named Customer_Sequence that gets incremented by 3 and use it for

inserting Custid values in Customer table. [10]

i. Develop a function named Customer_Loan which accepts Loanid as input and displays

Custid, CustName and loan_amount. [20]

```
Create procedure CustLoan_sp (Lnid in Number )
as begin
  Select t1.Custid, t1.CustName, t2.loan_amount from Customer t1, Loan t2 where t1.Custid =
t2.Custid; and t2.loanid = Lnid;
End ;
```

Exercise – 19

Consider the following relational schema for a Books Ordering database application:

Books (isbn, title, author, stock_qty, price, pub_year)

Customers (cust_id, cust_name, address)

Orders (order_no, cust_id, order_date) where cust_id refs

Customers(cust_id)

Order_list (order_no, isbn, qty, ship_date) where order_no refs

Orders(order_no), isbn refs Books (isbn)

a. The primary keys are underlined. Identify the foreign keys and draw schema diagram [5]

b. Create the above mentioned tables and populate the tables [20]

Note: Read all questions and populate values accordingly.

```
Create Table Books (isbn Number, title Varchar2(20), author Varchar2(20), stock_qty Number, price
Number, pub_year Number, Primary Key(isbn));

Create Table Customers (cust_id Number, cust_name Varchar2(20), address Varchar2(20));

Create Table Orders (order_no Number, cust_id Number, order_date Date, Primary Key(Orderr_no));

Create Table Order_list (order_no Number, isbn Number , qty Number, ship_date Date);
```

c. Include the constraint on Cust_id that it starts with letter 'C'. [5]

```
Alter Table Customer add constraint custidx Check(Custid like 'C%');
```

d. Display the custid and Custname who have ordered more than 3 books on the same date

[10]

```
Select t1.Custid, t1.Name from Cutomer t1, Orders t2, Order_list t3 where t1.Custid = T2.custid and
t2.orderno = t3.orderno group by t2.order_date.
```

e. Display the CustId and CustName who have ordered very few number of books. [10]

f. Create a view that keeps track of books that are ordered on 05-NOV-2014. Display isbn,

title, author, order_no, quantity and order_date. [20]

```
Create View Books_Order as
Select t1.Custid, t1.Name from Cutomer t1, Orders t2, Order_list t3 where t1.Custid = T2.custid and
t2.orderno = t3.orderno and  t2.order_date = '11/05/2014'
```

g. Create a procedure named Books_Ordered which outputs the customer name, book title and quantity ordered for the given order number [20]

h. In the above created procedure include exception to display "No such Order Number" if incorrect order number is given. [10]

```
Create procedure Books_Ordered (OrderNo in Number )
as begin
  Select t1.Custid, t1.CustName, t2.title, t3.qty from Customer t1, Books t2, Order_list t3 where
t1.Custid = t2.Custid; and t2.orderid = t3.orderid and t3.orderid = OrderNo;

  IF (if not exist(Select Order_no from Loan where Order_no = OrderNo) ) Then
        dbms_output.put_line('No Such Order_no Exist ')
  END IF;
End ;
```

Exercise – 20

Consider the following relational schema for Products Order database application:
Products (p_id, p_name, retail_price, qty_on_hand)
Orders (order_id, order_date)
Order_details (order_number, product_number, qty_ordered)
Where: order_number references order_id
product_number references p_id

a. The primary keys are underlined. Identify the foreign keys and draw schema diagram [5]

b. Create the above mentioned tables and populate the tables [20]

Note: Read all questions and populate values accordingly.

```
Create Table Products (p_id Number , p_name Varcahr2(20), retail_price Number , qty_on_hand
Number, Primary Key(P_id)) ;
Create Table Orders (order_id Number , order_date Date, Primary Key(Order_id));
Create Table Order_details (order_number Number, productNumber Number, qty_ordered Number,
primary key(order_number)) ;
```

c. Include the constraint on orderid that it starts with letter 'O'. [5]

```
Alter table order add constraint orderconstraint check (ordered like 'O%');
```

d. Display the ProdID and the sum of quantity ordered for each product. [10]

```
Select t1.prodid, SUM( t3. Qty) from Product t1, Order t2, Order_details t3 where t1. Proid = t2.proid
and t2.proid = t3.productNumber group by t3. productNumber
```

e. Create a view that keeps track of P_id, price, order_id, qty_ordered and ordered_date. [20]

```
Create View productSales as
Select t1.prodid, t1.price,t2.orderid,t2.order_date, t3.qty_ordered from Product t1, Orders t2,
Order_details t3 where t1. P_id = t3.productNumber and t2.orderid = t3.orderNumber group by
t3.proidproductNumber
```

g. Develop a procedure named Product_Orders that accepts a Product id or product number and displays all the order_details of the product. [10]

```
Create procedure Products_Orders (Prdtid in Number, PrdNumber in Number )
as begin
  Select t3.order_number, t3.productNumber, t3.qty_ordered  from Product t1, Orders t2,
Order_details t3 where t1. P_id = t3.productNumber and t2.orderid = t3.orderNumber
  and t1.productid = Prdtid and t3.prodcutNumber = PrdNumber
End ;
```

h. Create a database TRIGGER, which deletes the order from Orders table, AFTER the deletion of corresponding order_number in Order_details. [30]


```
create trigger Order_delete
after delete  on Order_details
for each row
begin
Delete * from Order where order_id = OrderNumber;
end;
```
