INFO7275 - Advanced Database Management Systems

Name: Vrushali Bandale

NUID: 001826854

Project Topic: Database for online shopping site (BestBuy)

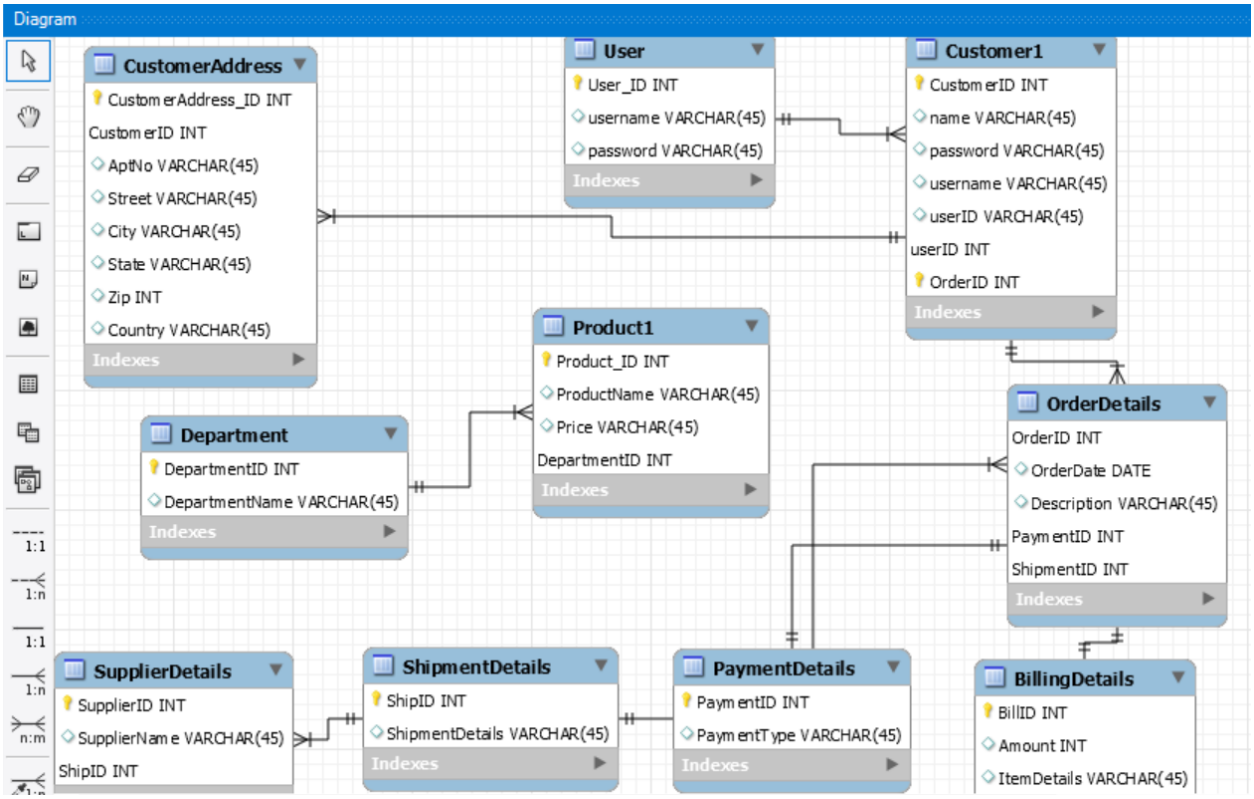
Problem Statement:

BestBuy is one of the world's largest company by revenue and is very popular for online shopping. Customers can place an order of Electronics equipment's, Laptops, Tablets etc. online and can also track the status of their orders. Thus, a very massive and important database must be maintained by BestBuy which stores the Product details, customer's details, customer order details and tracking details. I am going to design a database which will store all such information in Oracle.

Some important Entities:

Entities	Columns
Users	username, password
Customers details	username, password, userid
Order Details	order_id, order_date
Delivery/ Customer Address	apt_no, street,city,state,zip,country
Billing details	bill_id, amount, item details
Products	product_id, product_name, price
Department	dept_id, dept_name
Payment details	paym_id, paym_type
Supplier details	supp_id, supp_name
Shipment Details	ship_id, deliveryboy_name, supp_id

1. ER Diagram



2. Creating user

Username: PROJECT

Password: project

Pluggable database: pdbproject

Username: project

Password: project

```
Enter user-name: sys@orcl as sysdba
Enter password:

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> alter session set "_ORACLE_SCRIPT" = true;

Session altered.

SQL> create user c##project identified by project;

User created.

SQL> select username, common, oracle_maintained from all_users where username = 'C##PROJECT';

USERNAME
--------------------------------------------------------------------------------
COM O
--- -
C##PROJECT
YES Y


SQL> GRANT ALL PRIVILEGES TO C##PROJECT;

Grant succeeded.

SQL> conn C##PROJECT/project;
Connected.
SQL> alter system set pdb_file_name_convert = 'C:\Oracle19\oradata\ORCL\pdbseed\','C:\Oracle19\oradata\ORCL\pdbseed\' scope = both;

System altered.

SQL> create pluggable database pdbproject admin user project identified by project;
```

```
SQL> show pdbs;

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 ORCLPDB                        MOUNTED
         4 VRUPDB                         MOUNTED
         5 PDBPROJECT                     MOUNTED
SQL> alter pluggable database pdbproject open read write;

Pluggable database altered.

SQL> select status from v$instance;

STATUS
------------
OPEN

SQL> show pdbs;

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 ORCLPDB                        MOUNTED
         4 VRUPDB                         MOUNTED
         5 PDBPROJECT                     READ WRITE NO
SQL>
```

```
SQL> show pdbs;

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 ORCLPDB                        MOUNTED
         4 VRUPDB                         MOUNTED
         5 PDBPROJECT                     READ WRITE NO
SQL> disconnect
Disconnected from Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0
SQL> connect sys@orcl as sysdba
Enter password:
Connected.
SQL> connect project
Enter password:
ERROR:
ORA-01017: invalid username/password; logon denied


Warning: You are no longer connected to ORACLE.
SQL> disconnect;
SQL> connect sys@orcl as sysdba;
Enter password:
Connected.
SQL> conn sys@pdbproject as sysdba
Enter password:
Connected.
SQL>
```

1. External Tables

External Table allow oracle to query data that is stored outside the database in flat files. The ORACLE_LOADER driver can be used to access any data stored in any format that can be loaded by SQL*Loader. No DML can be performed on SQL Loader.

```
SQL> CREATE TABLE Cust_Details(
  2    CustID NUMBER,
  3    FirstName VARCHAR2(40),
  4    MiddleName VARCHAR2(20),
  5    LastName VARCHAR2(10)
  6   )
  7    ORGANIZATION EXTERNAL
  8  (TYPE ORACLE_LOADER
  9  DEFAULT DIRECTORY ext_tab_dir
 10  ACCESS PARAMETERS
 11  ( NOBADFILE
 12  FIELDS TERMINATED BY ','
 13  )
 14  LOCATION('demo.csv')
 15  )
 16  REJECT LIMIT UNLIMITED
 17  /

Table created.

SQL> DESC Cust_Details;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTID                                             NUMBER
 FIRSTNAME                                          VARCHAR2(40)
 MIDDLENAME                                         VARCHAR2(20)
 LASTNAME                                           VARCHAR2(10)

SQL>
```

- View: A view is a tailored presentation of data stored in the database. A view is a stored query. It can be used as a table just the difference is it does not store the data it just stores query definition.
- Relational View: It is basically a stored query, the output of which can be treated as if it were a table.

```
SQL> CREATE OR REPLACE VIEW Order_Details as
  2  select distinct c.name, OD.OrderID, OD.Description, PD.PaymentType, BD.Amount from
  3  Customer1 c inner join OrderDetails OD
  4  on c.OrderID = OD.OrderID
  5  inner join PaymentDetails PD
  6  on OD.PaymentID = PD.PaymentID
  7  inner join BillingDetails BD
  8  on OD.OrderID = BD.OrderID;

View created.
```

```
SQL> select * from Order_Details;

NAME                                 ORDERID DESCRIPTION                                        PAYMENTTYPE                                                AMOUNT
------------------------------------ ------- -------------------------------------------------- ---------------------------------------------------------- ----------
payal                                      2 iphone                                             Credit                                                        999
kevin                                      4 Speakers                                           Credit                                                        100
vrushali                                   1 ipad                                               COD                                                           399
john                                       3 Laptop                                             Debit                                                         892
```

- ❖ Inline View: Enable application developer to define views on the fly.

Select top 5 overpriced product

```
SQL>  select
  2  *
  3  from
  4  (
  5  select productID, ProductName, Price from Product1 order by price desc
  6  )
  7  where rownum<=5;

 PRODUCTID PRODUCTNAME                                          PRICE
---------- -------------------------------------------------- --------------------------------------------------
         2 Apple                                              999
         3 Dell                                               892
         1 Apple                                              399
         4 Sony                                               100

SQL>
```

- ❖ Materialized view: Allow the developer to precalculated view results and store those values enabling faster response time from the queries.

```
SQL>  Create Materialized view AdressDetails
  2   build immediate
  3   refresh on commit
  4   as
  5   select CustAddress_ID, AptNo, Street, City, State from CustomerAddress;

Materialized view created.

SQL> select * from AdressDetails;

CUSTADDRESS_ID APTNO                                  STREET             CITY                                               STATE
-------------- -------------------------------------- ------------------ -------------------------------------------------- --------------------
             1 28                                     St Stephen         Boston                                             MA
             2 67                                     St German          Boston                                             MA
             3 2                                      Mass Ave           Boston                                             MA
             4 6                                      Huntington Ave     Boston                                             MA

SQL>
```

5. Cursor: A cursor is a type of pointer built in PL/SQL for querying the database and retrieving a set of records and allowing developer to access the result set a row at a time. This lets programmer accomplish task that require procedural code to be performed on each record in result set individually.

```
SQL>  declare
  2   cursor custcursor (p_custid in number)
  3   is select *
  4   from Customer1 where CustomerID = p_custid;
  5   l_cust Customer1%rowtype;
  6   begin
  7   dbms_output.put_line('Getting details for Customer ID 1');
  8   open custcursor(1);
  9   loop
 10     fetch custcursor into l_cust;
 11     exit when custcursor%notfound;
 12     dbms_output.put('Customer ID'|| l_cust.customerID || 'is');
 13     dbms_output.put_line(l_cust.name);
 14     end loop;
 15     close custcursor;
 16
 17     dbms_output.put_line('Getting details for Customer ID 2');
 18   open custcursor(2);
 19   loop
 20     fetch custcursor into l_cust;
 21     exit when custcursor%notfound;
 22     dbms_output.put('Customer ID'|| l_cust.customerID || 'is');
 23     dbms_output.put_line(l_cust.name);
 24     end loop;
 25     close custcursor;
 26     end;
 27     /
Getting details for Customer ID 1
Customer ID1isvrushali
Getting details for Customer ID 2
Customer ID2ispayal

PL/SQL procedure successfully completed.

SQL>
```

6. Triggers: A trigger is a special type of stored procedure that automatically runs when an event occurs in the database server. DML triggers run when users try to modify data through data manipulation language(DML).

```
SQL>  CREATE OR REPLACE TRIGGER prod_audit
  2   AFTER UPDATE or DELETE on Product1
  3   DECLARE
  4  v_transaction VARCHAR2(10);
  5  BEGIN
  6  v_transaction:= CASE
  7  WHEN UPDATING THEN 'UPDATE'
  8  WHEN DELETING THEN 'DELETE'
  9  END;
 10  ------------ INSERT NEW ROW IN AUDIT TABLE ---------------------------
 11  INSERT INTO audits(table_name,transaction_name,by_user,transaction_date)
 12  VALUES('Products',v_transaction,USER, SYSDATE);
 13
 14  END;
 15  END;
 16
 17  update
 18  Product1
 19  set
 20  ProductName = 'Mac'
 21  where productID = 2;
 22
 23     /
 CREATE OR REPLACE TRIGGER prod_audit
                           *
ERROR at line 1:
ORA-04089: cannot create triggers on objects owned by SYS
```

7. Transactions using save point and rollback

A transaction is sequence of operations performed on a database as a single logical unit of work. The effect of all the SQL statements in the transactions can be either committed or rolled back.

```
SQL> CREATE TABLE t1(
  2  testcol number);

Table created.

SQL> DECLARE
  2  i INTEGER :=3;
  3  BEGIN
  4  INSERT INTO t1(testcol) VALUES (10/i);
  5  SAVEPOINT A;
  6  i:=i-1;
  7  INSERT INTO t1(testcol) VALUES (10/i);
  8  i:=i-1;
  9  INSERT INTO t1(testcol) VALUES (10/i);
 10  i:=i-1;
 11  INSERT INTO t1(testcol) VALUES (10/i);
 12  i:=i-1;
 13  INSERT INTO t1(testcol) VALUES (10/i);
 14  COMMIT;
 15  EXCEPTION
 16  WHEN ZERO_DIVIDE THEN
 17  ROLLBACK TO SAVEPOINT A;
 18  COMMIT;
 19  END testblock;
 20  /

PL/SQL procedure successfully completed.

SQL> select * from t1;

   TESTCOL
----------
3.33333333

SQL>
```

8. Index

An index is used to speed up the performance of the queries. It does this by reducing the number of database data pages that have to be viewed of scanned.

```
SQL> CREATE INDEX cust_index on Customer1(name);

Index created.

SQL>
```

9. Function: To get Address Details of the Customer

A function can be used as a part of SQL expression i.e we can use them with select/ update/ merge commands. The most important functionality of the function is that it must return a value.

```
SQL> CREATE OR REPLACE FUNCTION address_details (c_cust_id in number)
  2  RETURN VARCHAR2
  3  IS customer_address VARCHAR2(150);
  4  BEGIN
  5  SELECT 'Name-' ||Customer1.name|| 'Street -' || CustomerAddress.Street || 'City -' ||CustomerAddress.City || 'State -' || CustomerAddress.state ||
  6  'Country -' || CustomerAddress.Country into customer_address
  7  from Customer1, CustomerAddress where
  8  Customer1.customerID = CustomerAddress.customerID
  9  and customer1.customerID=c_cust_id;
 10  RETURN(customer_address);
 11  END address_details;
 12  /

Function created.

SQL> select address_details(1) as "Customer Address" FROM DUAL;

Customer Address
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------
Name-vrushaliStreet -St StephenCity -BostonState -MACountry -USA

SQL>
```

10. Procedure with cursor

A stored procedure is prepared SQL code that you can save so that code can be used over and over again. So, if you have a SQL query that you write repeatedly, save it as stored procedure and then just call it and execute it.

```
SQL> CREATE or REPLACE PROCEDURE cusrsor is
  2  cursor c1 is
  3  select CustomerID, name from Customer1 where CustomerID = 1;
  4  vcust_id Customer1.customerID%type;
  5  vname Customer1.customerID%type;
  6  begin
  7  open c1;
  8  loop
  9  fetch c1 into vcust_id, vname;
 10  exit when c1%notfound;
 11  dbms_output.put_line(vcust_id || ' ' || vname);
 12  end loop;
 13  close c1;
 14  end;
 15  /

Procedure created.
```

11. Reference Cursor using Records

A REF CURSOR is PL/SQL datatype whose value is the memory address of the query work area on the database. A REF Cursor is a pointer or a handle to a result set on a database. REF Cursor are represented through OracleRefCursor.

```
SQL>   declare
  2  type prod_dept_rec is record(
  3  ProductID number,
  4  ProductName varchar2(66),
  5  DepartmentName varchar2(37)
  6  );
  7
  8  type prod_dept_refcur_type is ref cursor
  9  return prod_dept_rec;
 10
 11  product_refcur prod_dept_refcur_type;
 12
 13  prod_dept prod_dept_rec;
 14  begin
 15  open product_refcur for
 16  select p.ProductID,
 17     p.ProductName,
 18     d.DepartmentName
 19  from Product1 p, Department d
 20  where p.DepartmentID = d.DepartmentID
 21  and rownum < 5
 22  order by p.ProductID;
 23
 24  fetch product_refcur into prod_dept;
 25  while product_refcur%FOUND loop
 26  dbms_output.put(prod_dept.ProductName|| '''s department is ');
 27  dbms_output.put_line(prod_dept.DepartmentName);
 28  fetch product_refcur into prod_dept;
 29  end loop;
 30  end;
 31  /
Apple's department is Tablet
Apple's department is Phone
Dell's department is Laptop
Sony's department is Speakers

PL/SQL procedure successfully completed.

SQL>
```

12. Pre-defined Exception

Oracle has predefined some common exception. These exceptions have a unique exception name and error number. These exceptions are already defined in ‘STANDARD’ package in oracle.

```
SQL>   declare
  2  e_amount BillingDetails%rowtype;
  3  New_Exception exception;
  4  begin
  5  e_amount.BillID := 2;
  6  e_amount.Amount := 'AS';
  7  insert into BillingDetails (BillID,Amount)
  8  values ( e_amount.BillID, e_amount.Amount );
  9  exception
 10  when VALUE_ERROR then
 11  dbms_output.put_line('We encountered the VALUE_ERROR exception');
 12  end;
 13  /
We encountered the VALUE_ERROR exception

PL/SQL procedure successfully completed.
```

Appendix

```
Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> alter session set "_ORACLE_SCRIPT" = true;

Session altered.

SQL> create user c##project identified by project;

User created.

SQL> select username, common, oracle_maintained from all_users where username = 'C##PROJECT';

USERNAME
--------------------------------------------------------------------------------
COM O
--- -
C##PROJECT
YES Y


SQL> GRANT ALL PRIVILEGES TO C##PROJECT;

Grant succeeded.

SQL> conn C##PROJECT/project;
```

Connected.

SQL> alter system set pdb_file_name_convert =
'C:\Oracle19\oradata\ORCL\pdbseed\','C:\Oracle19\oradata\ORCL\pdbseed\' scope = both;

System altered.

SQL> show pdbs;

CON_ID	CON_NAME	OPEN MODE	RESTRICTED
2	PDB$SEED	READ ONLY	NO
3	ORCLPDB	MOUNTED	
4	VRUPDB	MOUNTED	
5	PDBPROJECT	MOUNTED	

SQL> alter pluggable database pdbproject open read write;

Pluggable database altered.

SQL> select status from v$instance;

STATUS

OPEN

SQL> show pdbs;

CON_ID	CON_NAME	OPEN MODE	RESTRICTED

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

2 PDB$SEED READ ONLY NO

3 ORCLPDB MOUNTED

4 VRUPDB MOUNTED

5 PDBPROJECT READ WRITE NO

SQL>

SQL> show pdbs;

CON_ID CON_NAME OPEN MODE RESTRICTED

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

2 PDB$SEED READ ONLY NO

3 ORCLPDB MOUNTED

4 VRUPDB MOUNTED

5 PDBPROJECT READ WRITE NO

SQL> disconnect

Disconnected from Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production

Version 19.3.0.0.0

SQL> connect sys@orcl as sysdba

Enter password:

Connected.

SQL> connect project

Enter password:

ERROR:

ORA-01017: invalid username/password; logon denied

```
Warning: You are no longer connected to ORACLE.
SQL> disconnect;
SQL> connect sys@orcl as sysdba;
Enter password:
Connected.
SQL> conn sys@pdbproject as sysdba
Enter password:
Connected.
SQL>
```

```
CREATE TABLE User1(
  userID NUMBER,
  username VARCHAR2(20),
  password VARCHAR2(20),
  CONSTRAINT User_PK PRIMARY KEY (userID));

  CREATE TABLE Customer1(
  CustomerID NUMBER,
  name VARCHAR2(40),
  password VARCHAR2(20),
  username VARCHAR2(10),
  userID NUMBER,
  OrderID NUMBER,
  CONSTRAINT Customer_PK1 PRIMARY KEY (CustomerID));

  ALTER TABLE Customer1
  ADD CONSTRAINT Customer_FK2
```

```
FOREIGN KEY (userID)
REFERENCES User1(userID);

 ALTER TABLE Customer1
ADD CONSTRAINT Customer_FK3
FOREIGN KEY (OrderID)
REFERENCES OrderDetails(OrderID);

CREATE TABLE CustomerAddress(
CustAddress_ID NUMBER,
CustomerID NUMBER,
AptNo VARCHAR2(40),
Street VARCHAR2(50),
City VARCHAR2(50),
State VARCHAR2(20),
Zip NUMBER(6),
Country VARCHAR2(20),
CONSTRAINT Costomer_PK1 PRIMARY KEY (CustAddress_ID));


ALTER TABLE CustomerAddress
ADD CONSTRAINT Customer_FK1
FOREIGN KEY (CustomerID)
REFERENCES Customer(CustomerID);

 ALTER TABLE CustomerAddress
ADD CONSTRAINT Order_FK1
FOREIGN KEY (OrderID)
REFERENCES Order(OrderID);
```

```
CREATE TABLE OrderDetails(
OrderID NUMBER,
OrderDate DATE,
Description VARCHAR2(50),
PaymentID NUMBER,
CONSTRAINT Order_PK PRIMARY KEY (OrderID));

ALTER TABLE OrderDetails
ADD CONSTRAINT Order_FK
FOREIGN KEY (PaymentID)
REFERENCES PaymentDetails(PaymentID);



CREATE TABLE BillingDetails(
BillID NUMBER,
Amount NUMBER,
ItemDetails VARCHAR2(50),
OrderID NUMBER,
CONSTRAINT Bill_PK PRIMARY KEY (BillID));

ALTER TABLE BillingDetails
ADD CONSTRAINT Order1_FK
FOREIGN KEY (OrderID)
REFERENCES OrderDetails(OrderID);

CREATE TABLE Product1(
ProductID NUMBER,
```

```
ProductName VARCHAR2(50),
Price VARCHAR2(50),
DepartmentID NUMBER,
CONSTRAINT Product_PK4 PRIMARY KEY (ProductID));

ALTER TABLE Product
ADD CONSTRAINT Product_FK4
FOREIGN KEY (ProductID)
REFERENCES Product(ProductID);

ALTER TABLE Product1
ADD CONSTRAINT Department_FK4
FOREIGN KEY (DepartmentID)
REFERENCES Department(DepartmentID);

CREATE TABLE Department(
DepartmentID NUMBER,
DepartmentName VARCHAR2(50),
CONSTRAINT Department_PK PRIMARY KEY (DepartmentID));

CREATE TABLE PaymentDetails(
PaymentID NUMBER,
PaymentType VARCHAR2(50),
CONSTRAINT Payment_PK PRIMARY KEY (PaymentID));

CREATE TABLE SupplierDetails(
SupplierID NUMBER,
SupplierName VARCHAR2(50),
ShipID NUMBER,
```

```
CONSTRAINT Supplier_PK PRIMARY KEY (SupplierID));

ALTER TABLE SupplierDetails
ADD CONSTRAINT Ship_FK
FOREIGN KEY (ShipID)
REFERENCES ShipmentDetails(ShipID);

CREATE TABLE ShipmentDetails(
ShipID NUMBER,
ShipmentDetails VARCHAR2(50),
CONSTRAINT Ship_PK PRIMARY KEY (ShipID));

INSERT INTO User1 VALUES (1,'vrushali','admin');
INSERT INTO User1 VALUES (2,'payal','secret');
INSERT INTO User1 VALUES (3,'john','login');
INSERT INTO User1 VALUES (4,'kevin','abcd');

INSERT INTO Customer1 VALUES (1, 'vrushali','admin','vrushali', 1,1);
INSERT INTO Customer1 VALUES (2, 'payal','secret','payal', 2,2);
INSERT INTO Customer1 VALUES (3, 'john','login','john', 3,3);
INSERT INTO Customer1 VALUES (4, 'kevin','abcd', 'kevin', 4,4);

INSERT INTO CustomerAddress VALUES (1,1, 28, 'St Stephen', 'Boston', 'MA',02115,'USA' );
INSERT INTO CustomerAddress VALUES (2,2, 67, 'St German', 'Boston', 'MA',02115,'USA' );
INSERT INTO CustomerAddress VALUES (3,3, 2, 'Mass Ave', 'Boston', 'MA',02115,'USA' );
INSERT INTO CustomerAddress VALUES (4,4, 6, 'Huntington Ave', 'Boston', 'MA',02115,'USA' );

INSERT INTO OrderDetails VALUES (1, TO_DATE('22/April/2011 8:30:00AM','DD/MON/YY
HH:MI:SSAM'), 'ipad', 1);
```

```
INSERT INTO OrderDetails VALUES (2, TO_DATE('05/June/2014 9:22:00PM','DD/MON/YY HH:MI:SSAM'), 'iphone', 2);

INSERT INTO OrderDetails VALUES (3, TO_DATE('02/July/2018 11:42:22AM','DD/MON/YY HH:MI:SSAM'), 'Laptop', 3);

INSERT INTO OrderDetails VALUES (4, TO_DATE('01/April/2019 05:18:00PM','DD/MON/YY HH:MI:SSAM'), 'Speakers', 4);


INSERT INTO PaymentDetails VALUES (1,'COD');

INSERT INTO PaymentDetails VALUES (2,'Credit');

INSERT INTO PaymentDetails VALUES (3,'Debit');

INSERT INTO PaymentDetails VALUES (4,'Credit');


INSERT INTO BillingDetails VALUES (1, 399, 'Apple',1);

INSERT INTO BillingDetails VALUES (2, 999, 'Apple',2);

INSERT INTO BillingDetails VALUES (3, 892, 'Dell',3);

INSERT INTO BillingDetails VALUES (4, 100, 'Sony',4);


INSERT INTO Department VALUES (1, 'Tablet');

INSERT INTO Department VALUES (2, 'Phone');

INSERT INTO Department VALUES (3, 'Laptop');

INSERT INTO Department VALUES (4, 'Speakers');


INSERT INTO Product1 VALUES (1, 'Apple',399,1);

INSERT INTO Product1 VALUES (2, 'Apple',999,2);

INSERT INTO Product1 VALUES (3, 'Dell',892,3);

INSERT INTO Product1 VALUES (4, 'Sony',100,4);


INSERT INTO ShipmentDetails VALUES(1, 'USPS');

INSERT INTO ShipmentDetails VALUES(2, 'UPS');

INSERT INTO ShipmentDetails VALUES(3, 'newlogistics');
```

```
INSERT INTO ShipmentDetails VALUES(4, 'USPS');

INSERT INTO SupplierDetails VALUES(1, 'Best Buy', 1);
INSERT INTO SupplierDetails VALUES(2, 'Best Buy', 2);
INSERT INTO SupplierDetails VALUES(3, 'Best Buy', 3);
INSERT INTO SupplierDetails VALUES(4, 'Best Buy', 4);

ALTER TABLE CustomerDetails Add Column OrderID

Delete
 2  from Customer
 3  where OrderID is null;


CREATE TABLE Cust_Details(
  CustID NUMBER,
  FirstName VARCHAR2(40),
  MiddleName VARCHAR2(20),
  LastName VARCHAR2(10)
 )
  ORGANIZATION EXTERNAL
(TYPE ORACLE_LOADER
DEFAULT DIRECTORY ext_tab_dir
ACCESS PARAMETERS
( NOBADFILE
FIELDS TERMINATED BY ','
)
LOCATION('demo.csv')
)
```

```
REJECT LIMIT UNLIMITED
/

----------Relational View ----------

CREATE OR REPLACE VIEW Order_Details as
select distinct c.name, OD.OrderID, OD.Description, PD.PaymentType, BD.Amount from
Customer1 c inner join OrderDetails OD
on c.OrderID = OD.OrderID
inner join PaymentDetails PD
on OD.PaymentID = PD.PaymentID
inner join BillingDetails BD
on OD.OrderID = BD.OrderID

----------Inline View ----------

select
*
from
(
        select productID, ProductName, Price from Product1 order by price desc
        )
        where rownum<=5;

---------- Materialised View ----------

Create Materialized view AdressDetails
build immediate
refresh on commit
```

```
as
select CustAddress_ID, AptNo, Street, City, State from CustomerAddress;


---------- Cusrsor ----------
declare
cursor cust_cur (p_custid in number)
is select *
from Customer1 where CustomerID = p_custid;
l_cust Customer1%rowtype;
begin
dbms_output.put_line('Getting Employees for Customer ID 1');
open cust_cur(1);
loop
  fetch cust_cur into l_cust;
  exit when cust_cur%notfound;
  dbms_output.put('Customer ID'|| l_cust.customerID || 'is');
  dbms_output.put_line(l_cust.name);
  end loop;
  close cust_cur;

  dbms_output.put_line('Getting Employees for Customer ID 2');
open cust_cur(2);
loop
  fetch cust_cur into l_cust;
  exit when cust_cur%notfound;
  dbms_output.put('Customer ID'|| l_cust.customerID || 'is');
  dbms_output.put_line(l_cust.name);
  end loop;
  close cust_cur;
```

```
 end;
 /

 ---------- procedure ----------

 declare
create or replace UpdateAmount(ON OrderDetails.OrderID%type, AM BillingDetails.Amount%type)
is

CREATE TABLE audits(
auditID NUMBER generated BY DEFAULT as identity PRIMARY KEY,
table_name VARCHAR2(50),
transaction_name VARCHAR2(50),
by_user VARCHAR2(50),
transaction_date DATE
);

CREATE OR REPLACE TRIGGER prod_audit
AFTER UPDATE or DELETE on Product1
DECLARE
        v_transaction VARCHAR2(10);
BEGIN
v_transaction:= CASE
WHEN UPDATING THEN 'UPDATE'
WHEN DELETING THEN 'DELETE'
END;
---------- INSERT NEW ROW IN AUDIT TABLE ----------
INSERT INTO audits(table_name,transaction_name,by_user,transaction_date)
VALUES('Products',v_transaction,USER, SYSDATE);
```

```
END;
END;

update
Product1
set
ProductName = 'Mac'
where productID = 2;
```

```
CREATE TABLE t1(
testcol number);

DECLARE
i INTEGER :=3;
BEGIN
INSERT INTO t1(testcol) VALUES (10/i);
SAVEPOINT A;
i:=i-1;
INSERT INTO t1(testcol) VALUES (10/i);
i:=i-1;
INSERT INTO t1(testcol) VALUES (10/i);
i:=i-1;
INSERT INTO t1(testcol) VALUES (10/i);
i:=i-1;
INSERT INTO t1(testcol) VALUES (10/i);
COMMIT;
```

```
EXCEPTION
WHEN ZERO_DIVIDE THEN
ROLLBACK TO SAVEPOINT A;
COMMIT;
END testblock;



---------- Function to get complete address details of the customer----------


CREATE OR REPLACE FUNCTION address_details (c_cust_id in number)
RETURN VARCHAR2
IS customer_address VARCHAR2(150);
BEGIN
        SELECT 'Name-' ||Customer1.name|| 'Street -' || CustomerAddress.Street || 'City -'
||CustomerAddress.City || 'State -' || CustomerAddress.state ||
        'Country -' || CustomerAddress.Country into customer_address
        from Customer1, CustomerAddress where
        Customer1.customerID = CustomerAddress.customerID
        and customer1.customerID=c_cust_id;
        RETURN(customer_address);
        END address_details;
        /



CREATE or REPLACE PROCEDURE UpdatePayment(name IN VARCHAR2)
IS
        cnumber NUMBER;


        cusrsor c1 is
```

```
        SELECT PaymentID FROM PaymentDetails
        WHERE PaymentType = name;
BEGIN
        open c1;
        fetch c1 into cnumber;

        if c1%notfound then
        cnumber := 123;
        end if;

        INSERT INTO PaymentDetails(PaymentID, PaymentType) VALUES (cnumber, name);
        commit;
        close c1;

EXCEPTION
        WHEN OTHERS THEN
        raise_application_error(-20001,'An error was encountered-' ||SQLCODE|| 'ERROR -' ||
SQLERRM);
END;
/

execute UpdatePayment(1);

Select * from PaymentDetails;

CREATE or REPLACE PROCEDURE cusrsor is
cursor c1 is
select CustomerID, name from Customer1 where CustomerID = 1;
vcust_id Customer1.customerID%type;
```

```
vname Customer1.customerID%type;
begin
open c1;
loop
fetch c1 into vcust_id, vname;
exit when c1%notfound;
dbms_output.put_line(vcust_id || ' ' || vname);
end loop;
close c1;
end;

-------------------------------- Package ------------------------------------------------
CREATE or REPLACE PACKAGE pk1 AS
 FUNCTION getProd_name(n_id NUMBER)
 RETURN VARCHAR2;

 FUNCTION get_price(n_id NUMBER)
 RETURN NUMBER;
END pk1;

CREATE or REPLACE PACKAGE BODY pk1 AS

FUNCTION getProd_name(n_id NUMBER) RETURN VARCHAR2 IS
v_prodname VARCHAR2(50);
BEGIN
select ProductName into v_prodname from Product1 where productID = n_id;
RETURN v_prodname;
EXCEPTION
WHEN NO_DATA_FOUND THEN
```

```
RETURN NULL;
WHEN TOO_MANY_ROWS THEN
FUNCTION get_price(n_id NUMBER) RETURN NUMBER IS
n_price NUMBER(8,2);
BEGIN
SELECT Price INTO n_price FROM Product1 WHERE productID = n_id;
RETURN n_price;

EXCEPTION
WHEN NO_DATA_FOUND THEN
RETURN NULL;
WHEN TOO_MANY_ROWS THEN
RETURN NULL;
END;
END pk1;
/

---------- Reference Cursor ----------

declare
type prod_dept_rec is record(
        ProductID number,
        ProductName varchar2(66),
        DepartmentName varchar2(37)
        );

        type prod_dept_refcur_type is ref cursor
                return prod_dept_rec;
```

```
        product_refcur prod_dept_refcur_type;

        prod_dept prod_dept_rec;
begin
        open product_refcur for
                select p.ProductID,
                        p.ProductName,
                        d.DepartmentName
                from Product1 p, Department d
                where p.DepartmentID = d.DepartmentID
                and rownum < 5
                order by p.ProductID;

        fetch product_refcur into prod_dept;
        while product_refcur%FOUND loop
                dbms_output.put(prod_dept.ProductName|| '''s department is ');
                dbms_output.put_line(prod_dept.DepartmentName);
                fetch product_refcur into prod_dept;
        end loop;
end;
/
 ------------------------------------------ Exception----------------------------------
 declare
        e_amount BillingDetails%rowtype;
        New_Exception exception;
begin
        e_amount.BillID := 2;
        e_amount.Amount := 'AS';
        insert into BillingDetails (BillID,Amount)
```

```
        values ( e_amount.BillID, e_amount.Amount );
exception
        when VALUE_ERROR then
                dbms_output.put_line('We encountered the VALUE_ERROR exception');
end;
/
```