DMS Project

Game Store Database Management

Name	PRN	Seat No.
➢ Jay Prajapati	8021075028	453056
➢ Farhan Shaikh	8021075268	453067
➢ Vyom Vasava	8021077293	453079
➢ Bhautik Jani	8021076751	453019

Index :-

Description:

This database manages data of an online game store. In which details about games, their developers, users, offers, discount coupons, reviews and purchases are stored in the tables. Various functions and procedures to utilize the database are also shown below.

- User_purchase_total function finds total amount of purchases done by the user.
- Generate_coupon function generates a discount coupon for a user which provides discount for the games developed by a particular developer.
- Calculate_amount function calculates total amount to be paid for purchase of given game under given conditions (applied offers/coupons).
- Most_played_genre function finds out the genre which is most purchased by given user.
- Make_purchase is a procedure that makes the purchase of given game by given user along with calculating price for the purchase under effects of any offers or coupon discounts and inserts its data into appropriate tables.
- Discount_details procedure lists games that are available at a discounted price by means of offers and/or coupons for a user.
- Coup_disc_details and offer_disc_details are procedures which provide details about games which are available at discount by a given coupon or given offer respectively.
- Get_games_by_genre procedure finds games that belong to a given genre.
- Get_review procedure gathers reviews of a particular game for user.
- Delete_user procedure deletes all data related to a user which is to be deleted.
- Get_games_upto_budget procedure gives details about games which are available in user's budget.
- Along with above, a few functions are also useful in executing other procedures or functions.
- Triggers on purchase table are there which update number of downloads in games table, stop user from buying games which they aren't old enough to play, remove coupons which are once used in making purchase. Trigger on review table to update ratings of a game whenever a new review is received.

Tables :-

Tables of this database are given below with their create statements to describe their construction along with various constraints used to manage sanity of the data.

- Developer: This table contains details of the developers who produce games.

```
create table Developer(
    Developer_id varchar2(10) primary key,
    Name varchar2(40) not null,
    e_mail varchar2(50) check (e_mail like '_%@_%.com'),
    website varchar2(100) check (website like '_%.com')
);
```

- Games: This table contains data of the games like it's genre, release year, it's ratings, price, and reference to their developer through a foreign key.

```
create table Games (
     game_id varchar2(10) primary key,
     Name varchar2(30) not null,
     Genre varchar2(15) not null,
     Rating number(3,1),
     Total_Downloads number(10),
     Release_yr number(4) check (release_yr > 1900),
     Min_age number(2),
     Price number(7,2),
     Developer varchar2(10) constraint fk_1 references
     Developer(Developer_id) on delete cascade
);
```

- User_table: This table contains details about users who are using the online store to purchase games.

```
create table User_table (
    user_id varchar2(30) primary key,
    e_mail varchar2(50) check (e_mail like '_%@_%.com') not null,
    password varchar2(20) NOT NULL,
    age number(3) NOT NULL
);
```

- Purchase: This table contains details about purchase made by users, these records contain data of which game was purchased by which user since it is a weak entity it has a primary key which is combination of foreign keys of the strong entities on which it depends along with a candidate key of it's own to identify records independently.

```
create table purchase (
     purchase_id varchar2(12),
     game_id varchar2(10) references games(game_id) on delete cascade,
     user_id varchar2(30) references user_table(user_id) on delete
     cascade,
     primary key(purchase_id,game_id,user_id)
);
```

- Discount_coupons: This table contains details about discount coupons such as how much discount it provides, Which user owns it And it's expiry date.

```
create table discount_coupons (

coupon_id varchar2(12) primary key,
user_id varchar2(30) references user_table(user_id) not null on delete
cascade,
discount number(4,2) not null,
valid_till date not null
);
```

- Coupon_dev: This table contains data about relation between coupons and developers. Each coupon is related with a developer. And it provides discount on games which are developed by that developer.

```
create table coupon_dev (

 coupon_id varchar2(12) references discount_coupons(coupon_id) on delete
 cascade,
 developer_id varchar2(10) references developer(developer_id) on delete
 cascade,
 primary key (coupon_id, developer_id)
);
```

- Applied_discounts: This table stores details about offers and coupons which were applied for a particular purchase for purchase history.

```
create table applied_discounts (

purchase_id varchar2(12) references purchase(purchase_id) primary key on
delete cascade,
coupon_id varchar2(12) references discount_coupons(coupon_id) on delete
cascade,
offer_id varchar2(10) references offers(offer_id) on delete cascade,
amount number(7,2)
);
```

- Review: This table contains user reviews.

```
create table review (
   user_id varchar2(30) references user_table(user_id) on delete cascade,
   game_id varchar2(10) references Games(game_id) on delete cascade,
   Rating number(2) check (rating between 0 and 10) not null,
   review varchar2(100),
   constraint pk_rev primary key(user_id,game_id)
);
```

- Offers: This table contains details about offers, discount it provides, and it's validity.

```
create table Offers (
    offer_id varchar2(10) primary key,
    discount number(4,2),
    offer_start_dt date,
    offer_end_dt date,
    constraint ck_offr_dt check (offer_end_dt > offer_start_dt)
);
```

- Offer_games_list: This table contains data about relation between offers and games i.e. which offers are applicable on which games.

```
create table offer_games_list (
    offer_id varchar2(10) references offers(offer_id) on delete cascade,
    game_id varchar2(10) references games(game_id) on delete cascade,
    constraint pk_offrs primary key (offer_id,game_id)
);
```

Cardinalities:

Below are the cardinalities of relations between different entities.

➢ Developer – Games	:	1:M	
➢ Developer – Discount_Coupons	:	1:M	
➢ Games – User	:	M:N	
➢ Games – Offer	:	M:N	
➢ Games – Review	:	1:M	
➢ User – Review	:	1:M	
➢ (Games , User) – Review	:	1:1	(WEAK)
➢ (Games , User) – Purchase	:	1:1	(WEAK)
➢ Offer – Purchase	:	1:M	(WEAK)
➢ Discount_Coupon – Purchase	:	1:1	(WEAK)

Triggers:

1) update_game_rating : This trigger updates rating of games after any reviews about that games are inserted/updated/deleted from reviews table.

```
create or replace trigger update_game_rating
after insert or update or delete on review
for each row
begin
  update games
  set rating = (
    select avg(rating)
    from review
    where game_id = :new.game_id
  )
  where game_id = :new.game_id;
end;
/
```

2) check_user_age: This trigger checks upon purchase whether the user making purchase matches the minimum age requirement for that game.

```
create or replace trigger check_user_age
before insert on purchase
for each row
declare
  user_age number;
  game_min_age number;
begin
  select age into user_age from user_table where user_id =
:new.user_id;
  select min_age into game_min_age from games where game_id =
:new.game_id;

  if user_age < game_min_age then
```

```
        raise_application_error(-20001, 'user is not old enough to
    purchase this game');
      end if;
    end;
    /
```

3) This trigger removes coupons relation with developer once it is used so that this coupon can't be used again.

```
create or replace trigger remove_used_coupon
after insert on applied_discounts
for each row
declare
  coupon_id1 varchar2(12);
begin
  coupon_id1 := :new.coupon_id;

  if coupon_id1 is not null then
    delete from coupon_dev d
    where d.coupon_id = coupon_id1;
  end if;
end;
/
```

Functions:

1) purchase_total: This functon takes in as input user id and calculates the total amount of purchase done by that user.

```
create or replace function purchase_total(user_id in varchar2)
return number is
  total_purchase_amount number := 0;
begin

  insert into purchase_data(purchase_id, game_id, user_id,
purchase_amount)
  select p.purchase_id, p.game_id, p.user_id, d.amount
  from purchase p,applied_discounts d where d.purchase_id =
p.purchase_id
  and p.user_id = user_id;


  select sum(purchase_amount) into total_purchase_amount
  from purchase_data;

  return total_purchase_amount;
end;
/
```

2) generate_coupon : takes as input user to whom the coupon is given and discount which is provided by the coupon.

```
create or replace function generate_coupon(user_id in varchar2,disc
in number)
   return varchar2 is
   v_coupon_id varchar2(12);
     n number(10);
begin
     select count(*) into n from discount_coupons;

   --generate a unique coupon id
   select 'c' ||substr(user_id,1,4)|| lpad(n, 7, '0')
      into v_coupon_id
      from dual;
   --insert the new coupon into the discount_coupons table
   insert into discount_coupons (coupon_id, user_id, discount,
valid_till)
      values (v_coupon_id, user_id, disc, sysdate + 30);

   --return the generated coupon id
   return v_coupon_id;
end;
/
```

3) calculate_amount: This function calculates amount to be paid, takes as input user id, game id, coupon id, and offer which is applied.

```
 create or replace function calculate_amount(
    p_user_id in user_table.user_id%type,
    p_game_id in games.game_id%type,
    p_discount_coupon_id in discount_coupons.coupon_id%type default
null,
    p_offer_id in offers.offer_id%type default null
) return number
is
    v_game_price number(7,2);
    v_discount_amount number(7,2) := 0;
    v_final_price number(7,2);
begin
    select price
    into v_game_price
    from games
    where game_id = p_game_id;

    if p_discount_coupon_id is not null then
        select discount
        into v_discount_amount
        from discount_coupons
        where coupon_id = p_discount_coupon_id
          and user_id = p_user_id
          and valid_till >= sysdate;

        if v_discount_amount is null then
            raise_application_error(-20001, 'invalid coupon');
        end if;
    end if;

    if p_offer_id is not null then
        declare
            v_offer_start_dt date;
            v_offer_end_dt date;
        begin
            select offer_start_dt, offer_end_dt
            into v_offer_start_dt, v_offer_end_dt
            from offers
            where offer_id = p_offer_id;

            if v_offer_start_dt > sysdate or v_offer_end_dt <
sysdate then
                raise_application_error(-20002, 'invalid offer');
            end if;
        end;
    end if;

    if p_offer_id is not null then
        v_final_price := v_game_price * (1 - (select discount from
offers where offer_id = p_offer_id));
    else
        v_final_price := v_game_price;
    end if;

    if p_discount_coupon_id is not null then
        v_final_price := v_final_price * (1 - v_discount_amount);
    end if;

    return v_final_price;
exception
    when others then
        raise_application_error(-20000, 'error calculating amount
to pay');
end;
/
```

4) This function takes as input user id and finds which genre games user has purchased most.

```
create or replace function most_played_genre(user_id varchar2)
```

```
return varchar2
is
    genre varchar2(15);
begin
    select g.genre
    into genre
    from games g
    join purchase p on g.game_id = p.game_id
    where p.user_id = user_id
    group by g.genre
    order by count(*) desc
    fetch first 1 row only;

    return genre;
exception
    when no_data_found then
        return null;
end;
/
```

5) Some other functions which are used to implement procedures.

```
create or replace function find_dev
(c_id discount_coupons.coupon_id%type)
return varchar2 is
d_id developer.developer_id%type;
begin
  select developer_id into d_id from coupon_dev where coupon_id =
  c_id;
          return d_id;
  exception
    when no_data_found then return null;
  end;
  /

create or replace function find_offer
(g_id varchar2) return varchar2 is
  offr_id offers.offer_id%type;
begin
select ogl.offer_id into offr_id
    from offer_games_list ogl
    join offers o ON o.offer_id = ogl.offer_id
    where ogl.game_id = g_id
    and sysdate between o.offer_start_dt and o.offer_end_dt
    order by o.discount desc Fetch first 1 row only;
```

```
     return offr_id;
exception
when no_data_found then return null;
end;
/


create or replace function find_coup
(u_id varchar2, g_dev varchar2) return varchar2 is
    coup coupon_dev.coupon_id%type;
begin
    select c1.coupon_id into coup from discount_coupons c1 join
coupon_dev cd on c1.coupon_id = cd.coupon_id
    where c1.user_id = u_id and cd.developer_id = g_dev;

     return coup;
exception
    when no_data_found then return null;
end;
/
```

Procedures:

1. This procedure takes as input user id, game id and coupon id of coupon applied and makes purchase of given game by given user by determining price after considering discount from coupon and offers applicable.

```
create or replace procedure make_purchase
(u_id in user_table.user_id%type , g_id in games.game_id%type,
coup discount_coupons.coupon_id%type)
is

amnt number(7,2);
disc1 number (4,2);
disc2 number (4,2);
dev_id games.developer%type;
dev_id1 games.developer%type;
cursor cp is select * from discount_coupons;
r1 discount_coupons%rowtype;
p_id purchase.purchase_id%type;
offr_id offers.offer_id%type;
inv_coup Exception;
```

```
begin

select developer into dev_id from games where game_id = g_id;
offr_id := find_offer(g_id);

if offr_id is null then disc1 := 0;
else select o2.discount into disc1 from offers o2 where
offer_id = offr_id;
end if;

if coup is not null then

     for r2 in cp
    loop
    r1 := r2;
     if r1.coupon_id = coup then exit;
     end if;
    end loop;
     if r1.coupon_id <> coup then raise inv_coup;
     else
        dev_id1 := find_dev(coup);
          if dev_id1 is null then raise inv_coup;
          end if;
        if dev_id = dev_id1 and u_id = r1.user_id and sysdate
<= r1.valid_till
          then
     select discount into disc2 from discount_coupons where
coupon_id = coup;
          else raise inv_coup;
          end if;
     end if;
else disc2 := 0;
end if;

select price into amnt from games where game_id = g_id;

disc1 := (disc1 + disc2)/100;
amnt := amnt * (1 - disc1);

select count(*)+1 into p_id from purchase ;

p_id := 'p' || lpad(p_id,11,0);

insert into purchase values (p_id,g_id,u_id);
insert into applied_discounts (p_id,coup,offr_id,amnt);
commit;

Exception

when inv_coup then dbms_output.put_line('Invalid Coupon
applied..');
end;
/
```

2. This procedure takes as input user id to find which games are available at a discounted price considering coupons that user owns and offers applicable.

```
create or replace procedure discount_details
(u_id in varchar2)
is

off_disc number(4,2) :=0;
avail_offr offers.offer_id%type;
coup_disc number(4,2) :=0;
avail_coup coupon_dev.coupon_id%type;
g_dev developer.developer_id%type;
g_nm games.name%type;
disc_amnt number(7,2);
price number(7,2);
cursor g1 is select * from games;

begin

for r1 in g1
loop
coup_disc := 0; off_disc := 0;

select g.developer,g.price,g.name into g_dev,price,g_nm from
games g where g.game_id = r1.game_id;

avail_offr := find_offer(r1.game_id);
if avail_offr is not null then
select o.discount into off_disc from offers o where o.offer_id
= avail_offr;
end if;
```

```
avail_coup := find_coup(u_id,g_dev);
if avail_coup is not null then
select c.discount into coup_disc from discount_coupons c where
c.coupon_id = avail_coup;
end if;

if (coup_disc <> 0 or off_disc <> 0) then
     disc_amnt := price - (price * (coup_disc + off_disc)/100);

     insert into disc_games_T values
     (r1.game_id,g_nm,coup_disc,avail_coup,off_disc,disc_amnt);

commit;
end if;

end loop;
end;
/
```

3. This procedure takes as input offer id and displays details of games which the offer provides.

```
create or replace procedure offer_disc_details (offer_id1
varchar2)
is

r1 offers%rowtype;
offer_expired Exception;
offer_not_available_yet Exception;
cursor c1 is select * from offer_details;

begin

select o.* into r1 from offers o where o.offer_id = offer_id1;

if sysdate < r1.offer_start_dt then raise
offer_not_available_yet;
end if;
if sysdate > r1.offer_end_dt then raise offer_expired;
end if;

insert into offer_details (game_id,name,discount,disc_amnt)
select g.game_id,g.name,r1.discount,
g.price - (g.price * r1.discount / 100) from games g
```

```
where g.game_id in
(select game_id from offer_games_list ogl where ogl.offer_id =
r1.offer_id);

dbms_output.put_line('Given offer '||offer_id1||' will provide
you following benefits :');

for r2 in c1
loop
dbms_output.put_line('Game : '||r2.name||
' at Price:'||r2.disc_amnt||' ('||r2.discount||'% discount).');
end loop;



Exception

when no_data_found then dbms_output.put_line('Invalid
Offer..!!');

when offer_expired then dbms_output.put_line('This offer has
expired..!!');

when offer_not_available_yet then dbms_output.put_line('This
offer is not available yet..!!');
end;
/
```

4. This procedure takes as input coupon id and displays details of games which the coupon provides at discount.

```
create or replace procedure coupon_disc_details
(u_id in varchar2, coup varchar2) is
c_dev developer.developer_id%type;
inv_coup exception;
r1 discount_coupons%rowtype;
cursor c1 is select * from coup_details;

begin

select c.* into r1 from discount_coupons c
where c.coupon_id = coup;

    if(u_id <> r1.user_id)
```

```
        then raise inv_coup;
     end if;

select developer_id into c_dev from coupon_dev
where coupon_id = coup;

insert into coup_details (game_id,name,discount,disc_amnt)
(select g.game_id,g.name,r1.discount,
g.price - (g.price * r1.discount / 100)
from games g where developer = c_dev);

commit;

dbms_output.put_line('Coupon '||coup||' will provide you
following benefits :');

for r2 in c1
loop
dbms_output.put_line('Game :'||r2.name||
' at Price:'||r2.disc_amnt||' ('||r2.discount||'% discount).');
end loop;


EXCEPTION
    when no_data_found OR inv_coup
    then dbms_output.put_line('Invalid coupon..!!');
end;
/
```

5. This procedure takes as input a genre and displays details about games that belong to that genre which are available.

```
Temp(game_id_t , name_t , genre_t , release_yr_t , min_age_t ,
price_t , developer_t)

create or replace procedure get_games_by_genre
(p_genre Games.genre % type)
as
     cursor c1 is select * from temp;
     r1 c1 % rowtype;

begin
```

```
insert into temp(game_id_t , name_t , genre_t , release_yr_t ,
min_age_t , price_t , developer_t)
(select game_id , name , genre , release_yr , min_age , price ,
developer from Games where p_genre = genre);

open c1;
loop
    fetch c1 into r1;
    exit when c1%notfound;

    dbms_output.put_line('Game ID :-' || r1.game_id_t || 'Name
:-' || name_t ||'Price :-' || r1.price_t || 'Developer ID :-'
|| r1.developer_t);

end loop;
close c1;

exception
    when no_data_found
        then dbms_output.put_line('It seems there is no game
    under the requested genre.');
end;
/
```

6. This procedure takes as input id of a game and displays it's reviews.

```
review_game(user_id_g , game_id_g , rating_g , review_g)

create or replace procedure get_review(gid games.game_id%type)
as
    cursor c2 is select * from review_game;
    r2 c2 % rowtype;

begin

insert into review_game(user_id_g , game_id_g , rating_g ,
review_g)
(select user_id , game_id , rating , review from review where
game_id = gid);

for r2 in c2
loop
```

```
    exit when c2 % notfound;
dbms_output.put_line(' User ID :-' || r2.user_id_g || ' Rating
:- ' || rating_g ||' Review :-' || r2.review_g);
end loop;

exception

when no_data_found then dbms_output.put_line('There are
currently no reviews for the requested game.');

end;
/
```

7. This procedure takes as input a user id and deletes that user from database.

```
create or replace procedure delete_user
(uid user_table.user_id%type)
as
begin
    delete from user_table where user_id = uid;
    commit;
end;
/
```

8. This procedure takes as input budget of a user and displays details about games which are available in that budget.

```
create or replace procedure get_games_upto_budget
(budget in games.price % type)
as

cursor cgames is select * from games where price <= budget;
r1 cgames% rowtype;

begin

dbms_output.put_line('The games under the price ' || budget ||
' are shown below:');

open cgames;
```

```
loop

fetch cgames into r1
exit when cgames % notfound;


dbms_output.put_line
('Game ID: ' || r1.game_id || ', Name: ' || r1.name || ',
Genre: ' || r1.genre || ', Rating: ' || r1.rating || ',
Downloads: ' || r1.total_downloads || ', Release Year: ' ||
r1.release_yr || ', Minimum Age: ' || r1.min_age || ', Price: '
|| r1.price || ', Developer: ' || r1.developer);

end loop;
close cgames;
end;
/
```