1) Write a **function** that returns the DVD ID of the next in stock DVD in the customer’s movie list (rental queue). The function should take as an IN parameter the customer ID and should return the DVD ID as the function value. The function should return the first title (the DVD ID) in the customer’s movie list that is in stock. If the movie list is empty or none of the titles are in stock, then the function should return NULL. Additionally, the function should perform error handling to check that the customer’s account balance is not negative.

ALTER TABLE member

ADD memberaccountbalance NUMBER DEFAULT 0;

Paste in a screenshot of the results running your code.

UPDATE MEMBER

SET MEMBERACCOUNTBALANCE = -10

WHERE MEMBERID = 1

CREATE OR REPLACE FUNCTION NEXTDVDINSTOCK (P\_MEMBERID IN NUMBER)

RETURN NUMBER

IS nextdvdid NUMBER;

m\_accountbalance NUMBER;

BEGIN

nextdvdid := NULL;

m\_accountbalance := NULL;

SELECT memberaccountbalance

INTO m\_accountbalance

FROM member

WHERE memberid = P\_MEMBERID;

IF m\_accountbalance < 0 THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Member''s account balance is negative.');

END IF;

SELECT d.dvdid --, d.dvdtitle, d.dvdquantityonhand, q.dateaddedinqueue

INTO nextdvdid

FROM DVD d

JOIN rental r ON d.dvdid = r.dvdid

JOIN rentalqueue q ON r.memberid = q.memberid

AND d.dvdquantityonhand != 0 -- filter so we have atleast 1 dvd avaiable

AND d.dvdquantityonhand IS NOT NULL

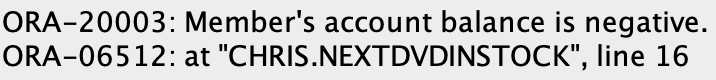
AND ROWNUM = 1

ORDER BY q.dateaddedinqueue;

RETURN nextdvdid;

END NEXTDVDINSTOCK;

SELECT nextdvdinstock(1) from dual;



SELECT nextdvdinstock(2) from dual;



2) Write a **function** that returns the number of additional DVDs that a customer may receive before they reach the limits of their contract. There are two limits for each plan, (1) how many can be rented at a time, and (2) how many can be rented per month. The function should take the customer ID as an IN parameter and should return the number of DVDs through its integer function value which the customer can rent based on their current rental activity and their plan.

CREATE OR REPLACE FUNCTION remainingdvdrentalsonplan (P\_MEMBERID IN NUMBER)

RETURN NUMBER

IS membershiplimit NUMBER;

dvdsalreadyrented NUMBER;

availabledvdstorent NUMBER;

startdate DATE;

enddate DATE;

BEGIN

membershiplimit := 0;

dvdsalreadyrented := 0;

startdate := NULL;

enddate:= NULL;

SELECT to\_date(trunc((to\_date(rentalshippeddate)),'month'))

INTO startdate

FROM rental r

JOIN member m ON r.memberid = m.memberid

WHERE m.memberid = P\_MEMBERID AND RentalReturnedDate IS NULL AND ROWNUM = 1;

SELECT to\_date(Last\_Day(MAX(RentalShippedDate)))

INTO enddate

FROM rental r

JOIN member m ON r.memberid = m.memberid

WHERE m.memberid = P\_MEMBERID AND RentalReturnedDate IS NULL AND ROWNUM = 1;

SELECT s.membershiplimitpermonth

INTO membershiplimit

FROM membership s

JOIN member m ON m.membershipid = s.membershipid

where m.memberid = P\_MEMBERID;

SELECT COUNT(r.dvdid)

INTO dvdsalreadyrented

FROM rental r

JOIN member m ON r.memberid = m.memberid

WHERE m.memberid = P\_MEMBERID

AND r.rentalshippeddate BETWEEN startdate AND enddate;

availabledvdstorent := membershiplimit - dvdsalreadyrented;

RETURN availabledvdstorent;

END remainingdvdrentalsonplan;

Paste in a screenshot of the results running your code.

SELECT remainingdvdrentalsonplan(1) from dual;



SELECT remainingdvdrentalsonplan(205) from dual;



3) Implement a trigger that prevents a customer from being shipped a DVD if they have reached their monthly limit for DVD rentals as per their membership contract using the function in from question 2 above.

CREATE OR REPLACE TRIGGER overdvdrentallimit

BEFORE INSERT OR UPDATE OF rentalshippeddate, memberid

ON rental

FOR EACH ROW

DECLARE availabletorent NUMBER;

BEGIN

SELECT remainingdvdrentalsonplan(:NEW.memberid)

INTO availabletorent

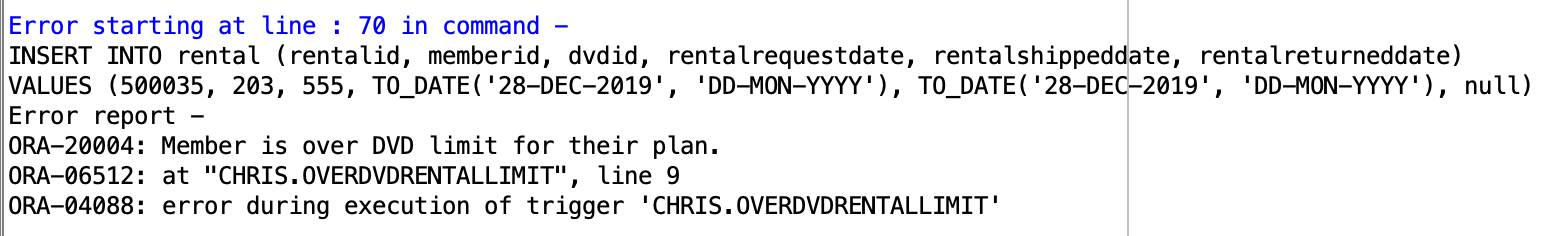
FROM dual;

IF availabletorent <= 0 THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Member is over DVD limit for their plan.');

END IF;

END;



4) Write a stored procedure that implements the processing when a DVD is returned in the mail from a customer and the next DVD is sent out. This processing should include recording that the DVD has been returned and should also determine the number of additional DVDs that should be mailed to the customer. Use the functions and stored procedures which you have already created to complete this transaction. At minimum your stored procedure must account for the following tasks.

* Customer returns a DVD or notes the DVD is lost in which case they are charged against their account.
* Initiate function from question 2 to return the number of additional dvds which can be rented.
* Initiate the function from question 1 to get a movie from the customer’s request list (rentalqueue) which is in stock.
* Initiate stored procedure from Advanced Programming part 2 which removes this DVD from the queue.
* Perform the rental of the above DVDs – since more than one DVD could be returned – the stored procedure needs to be able to handle dynamically the rental process of multiple DVDs returned with function #2 (remainingdvdrentalsonplan).
* Update all the DVD quantities accordingly. Note that you can implement this as part of this stored procedure – or as a separate trigger which is used by this stored procedure.

CREATE OR REPLACE PROCEDURE dvdreturnprocess (

P\_memberid NUMBER,

P\_dvdid NUMBER

)

IS

NumAvailableDvdsToRent NUMBER;

P\_NextDvdInStock NUMBER;

nextdvdtoreturn NUMBER;

BEGIN

IF P\_dvdid IS NULL THEN

UPDATE member

SET MEMBERACCOUNTBALANCE = MEMBERACCOUNTBALANCE - 10

WHERE memberid = P\_memberid;

END IF;

SELECT remainingdvdrentalsonplan(P\_memberid)

INTO NumAvailableDvdsToRent

FROM dual;

SELECT NextDvdInStock(P\_memberid)

INTO P\_NextDvdInStock

FROM dual;

DELETEMOVIE(P\_memberid, P\_dvdid);

UPDATE rental

SET RENTALRETURNEDDATE = SYSDATE

WHERE memberid = P\_memberid AND P\_dvdid = P\_dvdid;

UPDATE dvd

SET dvdquantityonhand = dvdquantityonhand + 1, dvdquantityonrent = dvdquantityonrent -1

WHERE dvdid = P\_dvdid;

nextdvdtoreturn := &enter\_next\_dvdid\_or\_0\_to\_finish;

WHILE &enter\_next\_dvdid\_or\_0\_to\_finish <> 0

LOOP

DELETEMOVIE(P\_memberid, nextdvdtoreturn);

UPDATE rental

SET RENTALRETURNEDDATE = (select SYSDATE from dual)

WHERE memberid = P\_memberid AND P\_dvdid = nextdvdtoreturn;

UPDATE dvdquantityonrent

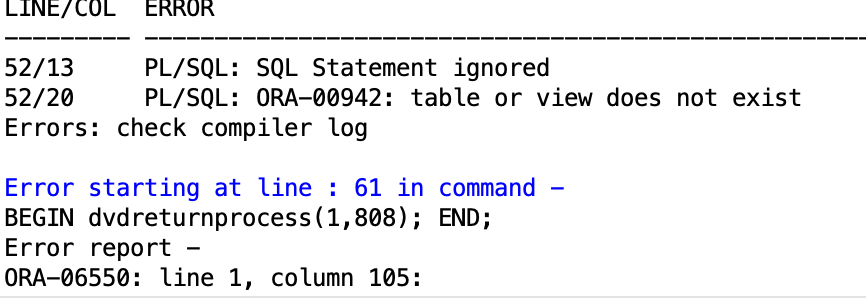
SET dvdquantityonhand = dvdquantityonhand + 1, dvdquantityonrent = dvdquantityonrent -1

WHERE dvdid = P\_dvdid;

nextdvdtoreturn := &enter\_next\_dvdid\_or\_0\_to\_finish;

END LOOP;

END;



Note: I kept getting table or view does not exist. Getting the input process from the user to work ended up being too difficult. However much of the rest of the code is still individually functional.

Extra Credit Questions

1) [Extra credit.] Write a stored procedure that checks that all of the data in the database satisfies all general database integrity constraints. This is an example of a stored procedure that may be run daily to verify that there have been no failures or compromise of the general database integrity constraints enforced by triggers and stored procedures. We do not need to check integrity constraints enforced by primary keys, foreign keys, check constraints, and not null constraints. An example might be that the in-stock quantity of DVDs is accurate based on the rental history. There might be other interesting and relevant scenarios.

CREATE TABLE DATACHECK\_HISTORY(

RUNDATE DATE,

DVDQUANTITYONRENTMISMATCHES NUMBER

);

CREATE OR REPLACE PROCEDURE DATACHECK AS

BEGIN

INSERT INTO DATACHECK\_HISTORY (RUNDATE, DVDQUANTITYONRENTMISMATCHES)

VALUES (

(SELECT SYSDATE FROM DUAL),

(SELECT COUNT(\*)

FROM

(

SELECT d.dvdid, d.dvdquantityonrent, COUNT(\*) as "RentalHistory"

FROM DVD d

JOIN RENTAL r ON r.dvdid = d.dvdid

WHERE RENTALRETURNEDDATE IS NULL

AND RENTALSHIPPEDDATE IS NOT NULL

GROUP BY d.dvdid, d.dvdquantityonrent

HAVING COUNT(\*) != d.dvdquantityonrent

ORDER BY 1

)

)

);

END DATACHECK;

EXECUTE DATACHECK();



2) [Extra credit.] Define and implement additional functionality using one or more stored procedures, functions, or triggers. Make sure to pick a relevant business scenario that has advanced level of complexity.

Scenario: Marketing wants to know who the most popular director is based off of rentals, so they can be sure to advertise that Netflix has those movies to customers and potential new customers.

CREATE OR REPLACE FUNCTION mostrenteddirector

RETURN VARCHAR2

IS

topdirector VARCHAR2(255);

BEGIN

SELECT personfirstname || ' ' || personlastname as "Director"

INTO topdirector

FROM

(SELECT mp.personfirstname || ' ' || mp.personlastname, mp.personfirstname, mp.personlastname, COUNT(r.dvdid) AS "TotalRentalsofFilms"

FROM moviepersonrole mpr

JOIN movieperson mp ON mp.personid = mpr.personid

JOIN rental r ON r.dvdid = mpr.dvdid

JOIN role ro ON ro.roleid = mpr.roleid

WHERE ro.rolename = 'Director'

GROUP BY mp.personfirstname || ' ' || mp.personlastname,mp.personfirstname, mp.personlastname

ORDER BY COUNT(r.dvdid) DESC)

WHERE Rownum = 1

;

RETURN topdirector;

END mostrenteddirector;

select mostrenteddirector() from dual;

