COMP23111

Academic year 2016 ~ 2017

Exercise Number: EX04

Name: Boyin Yang

Student ID: 10071239

```
-- [header]

--

-- COMP23111 Fundamentals of Databases

-- Exercise 04

-- by Boyin Yang, ID 10071239, login name mfbyaby3
-- [opening]

SET ECHO ON

SPOOL eclectic.txt

start /opt/info/courses/COMP23111/create-Eclectic-Ecommerce-tables.sql

start /opt/info/courses/COMP23111/populate-Eclectic-Ecommerce-tables.sql

 -- causes the SQL statements themselves to be spooled

 -- sends everything to <spoolfilename>



-- here you can set the SQL*Plus parameters, such as column width,

-- that will allow the script to produce readable answers in the spool

-- file



-- [body]
```

```
/* Task1 (a) */
create or replace view customer_name_with_carts as
      select distinct firstName, lastName
      from customerInfo, orderCartInfo
      where loginName = customerID;
select * from customer_name_with_carts;


/* Task1 (b) */
create or replace view item_in_inventory_need_reorder as
      select distinct inventoryItem.itemNum, code, belongsTo, qtyInstock
      from inventoryItem, itemType
      where qtyInstock<25 and inventoryItem.itemNum = itemType.itemNum
      order by itemNum;
select *from item_in_inventory_need_reorder;


/* Task1 (c) */
create or replace view customer_paid_in_each_order as
      select distinct loginName, firstName, lastName,
lineItems.orderCartId, sum(orderPrice) as totalPrice
      from customerInfo, lineItems, orderCartInfo
```

```
	where loginName=customerID and lineItems.orderCartId =
orderCartInfo.orderCartId
	group by loginName,firstName, lastName, lineItems.orderCartId
	order by orderCartId;
select * from customer_paid_in_each_order;


/* Task1 (d) */
create or replace view customer_paid_in_total as
	select distinct loginName, firstName, lastName, sum(totalPrice) as
totalPrices
	from customer_paid_in_each_order
	group by loginName, firstName, lastName
	order by loginName;
select * from customer_paid_in_total;


/* Task1 (e) */
create or replace view number_of_carts_per_customer as
	select distinct customerID, count(orderCartId) as num
	from orderCartInfo
	group by customerID
	order by customerID;
```

```
select customerID,
      case
      when num <= 2 then 'BR-1 satisfied'
      else 'BR-1 violated'
      end
      "OUTCOME"
      from number_of_carts_per_customer;


/* Task1 (f) */
select itemNum, itemColor, itemSize
from (
      select itemNum, itemColor,itemSize,
      case
      when typeNum <= 1 then 'BR-2 satisfied'
      else 'BR-2 violated'
      end
      "OUTCOME"
      from(
            select itemNum, itemColor, itemSize, count(*) as typeNum
            from inventoryItem
            group by itemNum,itemColor,itemSize
            order by itemNum
      )
)
where OUTCOME = 'BR-2 violated';


/* Task1 (g) */
select * from itemType;


create or replace trigger raise_insert_error
      after insert or update
      on itemType
      declare
      fourTimeMinPrice float;
      maxPrice float;
      begin
      select 4*min(price) into fourTimeMinPrice from itemType;
      select max(price) into maxPrice from itemType;
      if (maxPrice>fourTimeMinPrice) then
      raise_application_error
      (-20101,'Cannot insert/update because the new price is larger than
4 times of minimun price in table.');

      rollback;

      end if;

      end;

/



insert into itemType values ('C3', 'Google Autonomous Car', '(｡•ˇ‸ˇ•｡)',
79999.99, 'SF');

insert into itemType values ('C4', 'SpeaceX Falcon99', ' (´・ω・｀)',
7999999.99, 'SF');

update itemType set price = 100 where name = 'The Postlude';



start /opt/info/courses/COMP23111/drop-Eclectic-Ecommerce-tables.sql

SPOOL OFF



-- [footer]

--

-- End of Exercise 04 by Boyin Yang
```

Comments:

When trigger that using AFTER is triggered, data has not committed directly. So we can ROLLBACK before the end of that trigger if the new data is not we expect.