**CSCD 427/527 Lab 2 (20 points)**

**Get Ready:**

Now create your second database using:

create database *yourusername\_salesDB*;

Run script “salesDB.sql” to add new tables into the above database.

Once you have a database created, you need to highlight this database to make sure the following operations will be applied to this database.

**Part I: Views**

1. Use a *SELECT* statement to list all products including MFR\_ID, PRODUCT\_ID, and salesperson’s ID and the total quantity of sales (named as Total\_qty\_by\_salesperson). The result should include the products that have never been ordered. Hint: use an outer join. Include the SQL statement and a screen copy of the query result in your submission.

**select mfr\_id, product\_id, rep, coalesce(sum(qty), 0) as Total\_qty\_by\_salesperson**

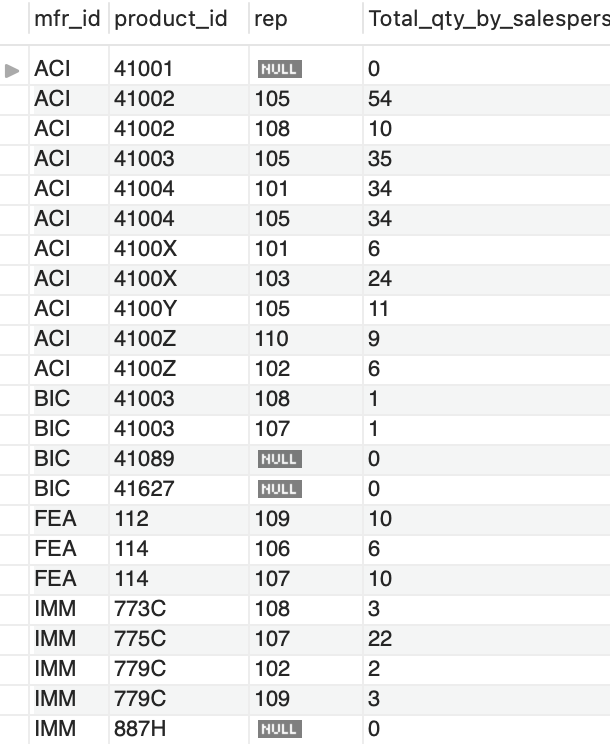
**from PRODUCTS left outer join ORDERS**

**on mfr\_id = mfr and product\_id = product**

**group by mfr\_id, product\_id, rep;**

**The result set includes 35 records.**

**Snapshot of the first few records:**

****

1. Write a SQL statement to create a view called *ProductSales* using the query from Question 1.

After you have successfully created the view, run the following query and copy the query result in your submission.

select \*

from ProductSales

order by mfr\_id, product\_id;

**create view ProductSales as**

**select mfr\_id, product\_id, rep, sum(qty) as Total\_QTY\_by\_salesperson**

**from PRODUCTS left outer join ORDERS**

**on mfr\_id = mfr and product\_id = product**

**group by mfr, product, rep;**

1. Insert a new row ('AAA', '44444', 'New Product', 100, 10)into *PRODUCTS* table; run the query from Question 2 again and copy the query result in your submission.

**insert into PRODUCTS values ('AAA', '44444', 'New Product', 100, 10);**

**A new role is added to the query result.**

1. Now let’s try to update the view *ProductSales* by running the following query:

UPDATE ProductSales

SET rep = 106

WHERE rep IS NULL;

Did the query succeed? If not, copy the error message, and explain why you received this error message.

**“The target table ProductSales of the UPDATE is not updatable”**

**Because the view was defined upon two tables.**

1. Use *ProductSales* to create another view called *ProductProfit* to show the MFR\_ID, PRODUCT\_ID, and total\_profit of the product. total\_profit = sum(Total\_qty\_by\_salesperson)\* PRICE \* 0.1 (this assumes the profit is 10%).

Then list the top 10 products based on *total\_profit*.

**create view ProductProfit as**

**select mfr\_id, product\_id, sum(Total\_qty\_by\_salesperson)\*price\*0.1 as Total\_profit**

**from ProductSales NATURAL JOIN PRODUCTS**

**group by mfr\_id, product\_id**

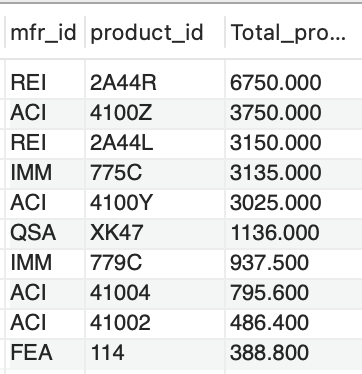
**The view includes 26 records.**

**select mfr\_id, product\_id, total\_profit**

**from ProductProfit**

**order by total\_profit desc**

**limit 10**

****

1. Update the view *ProductProfit* by changing the *total\_profit* of the product (MFR\_ID = ‘ACI’ and PRODUCT\_ID = 41001) to $100,000. Run the query. Did the query succeed? Why?

**update ProductProfit**

**set Total\_profit = 100000**

**where mfr\_id = 'ACI' and product\_id = 41001;**

**No, the query failed, because the view is not updatable.**

**Part II: Security**

As a database user, you have the privileges to access particular databases where you can create tables, insert into them, etc. Now you want to give permissions to other users to access your databases. For security reasons, you want to restrict access to your databases as much as possible, and only give privileges that are needed in order to accomplish some tasks. We'll accomplish this through the use of the SQL GRANT statement.

**GRANT <permissions>   
[ON <table(s)>]  
TO <user>**

**For this part of the lab, you will need a partner**. Please work with a classmate to accomplish and test the tasks below:

1. Ask your partner to log into MySQL Workbench using their username and password. Then ask your partner to run the following statement:

SELECT \* FROM *YourUsername\_salesDB.ORDERS;*

Did the query succeed? Why?

**Operation failed**

1. Write a GRANT statement to grant SELECT privileges on your *ORDERS* table to your partner. Copy the corresponding SQL statement in your submission.

Then ask your partner to run the query from Question 7 again (Your partner might need to refresh their schema to reflect all the recent changes). Did the query succeed? Why?

**Grant select on MyUsername\_salesDB.ORDERS to MyPartner**

**Operation succeeded**

1. Ask your partner to insert a new record into your *ORDERS* table with any arbitrary values. Did the operation succeed? Why?

**Operation failed because didn’t grant insert**

1. Now revoke the privilege that you previously granted to your partner. Copy the corresponding SQL statement in your submission.

Then ask your partner to run the query from Question 7 again (Your partner might need to refresh their schema to reflect all the recent changes). Did the query succeed? Why?

REVOKE SELECT ON ***YourUsername\_salesDB.ORDERS*** FROM *YourPartnersUsername.*

**Operation failed again.**