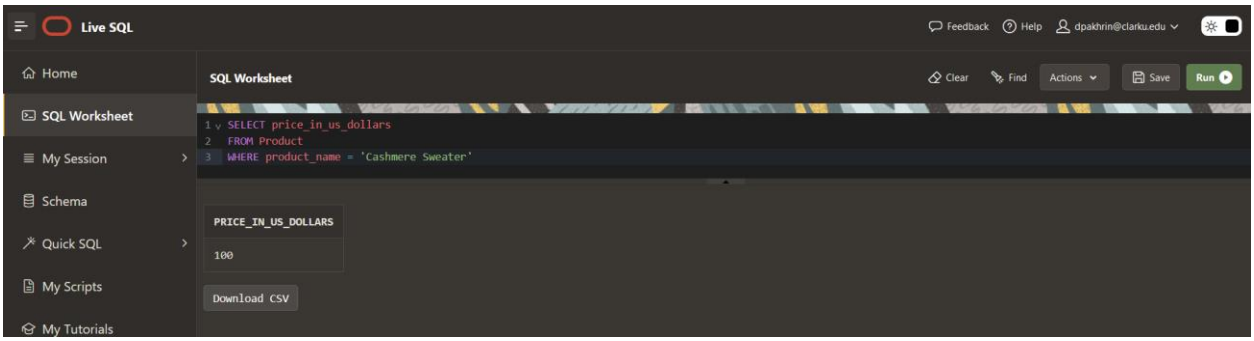
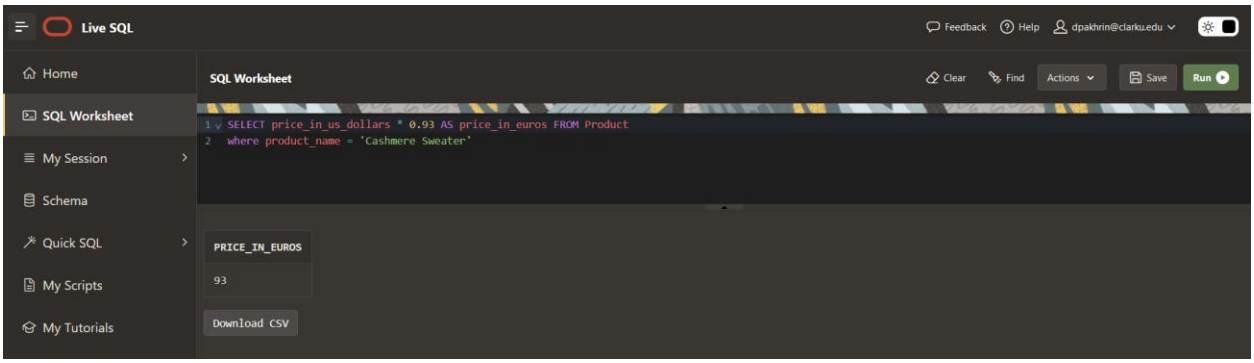
Lab 5 – Submission Sheet

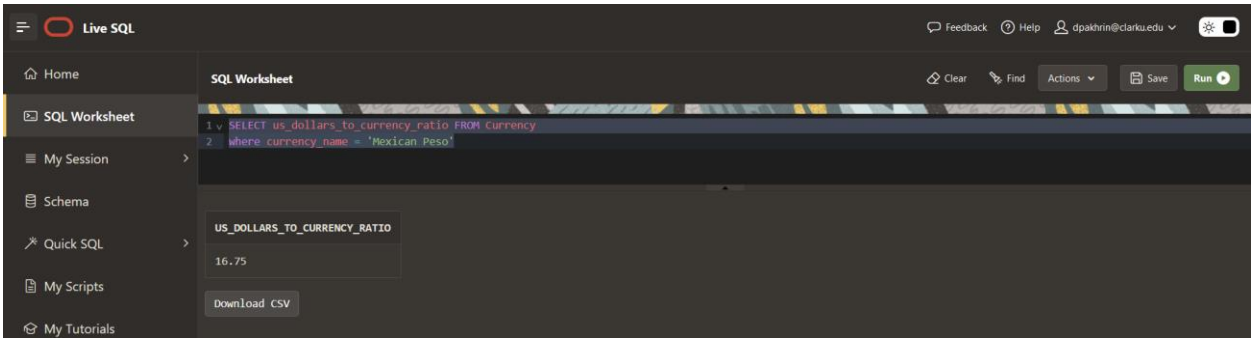
Section One

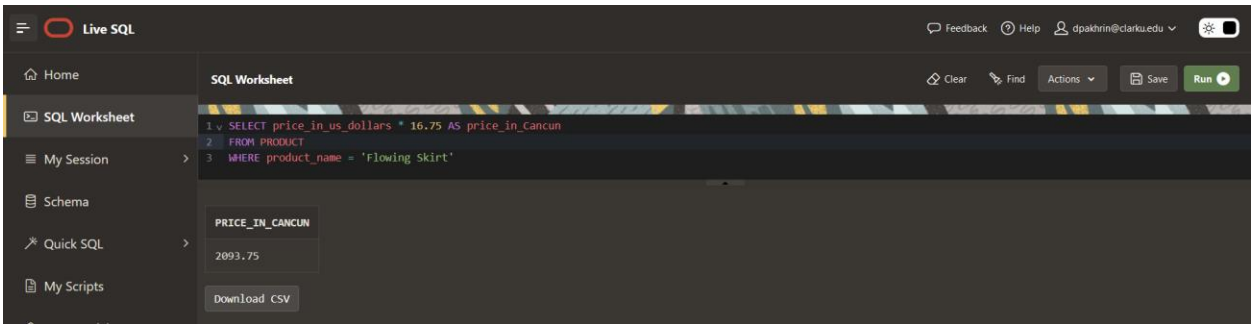
1. SELECT (Screen Shots)





2. SELECT (Screen Shots)





3.

a. EXPLANATION

Datatypes are the type of data to be recorded in the table's particular column. It ensures the validity of the values to be stored in that column.

For instance, the VARCHAR datatype allows only the string data to be stored in the column. To define the maximum size allowed for the column, we provide the size such that VARCHAR(255) that can store up to 255 characters.

If the value exceeds the defined size of the data, we receive value too large for column error.

b. CREATE TABLE (Screen Shot)

c. INSERT (Screen Shot)

4. EXPLANATION

Expression in a DBMS is a combination of operands and arithmetic signs resulting in an operation.

The DBMS strictly follows the precedence to determine the result. It follows the PEMDAS rule for the mathematical operation which means Parentheses, Exponents, Multiplication, Division, Addition, and Subtraction.

For example,

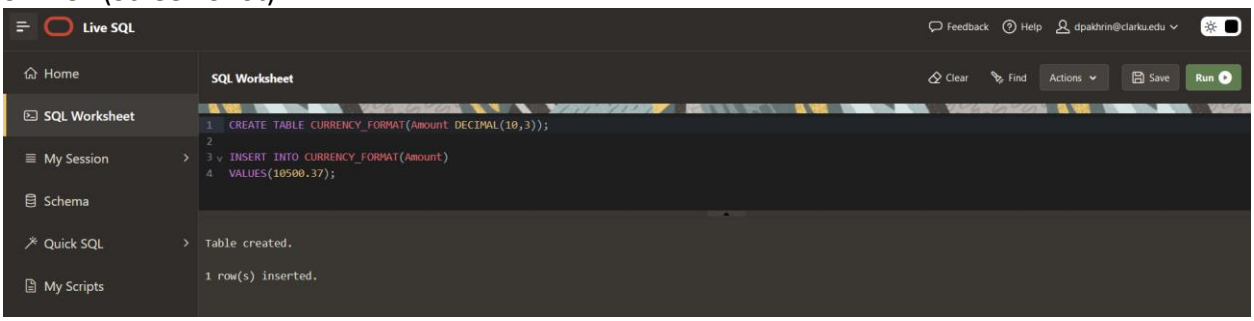
SELECT (5 + 3) * 2

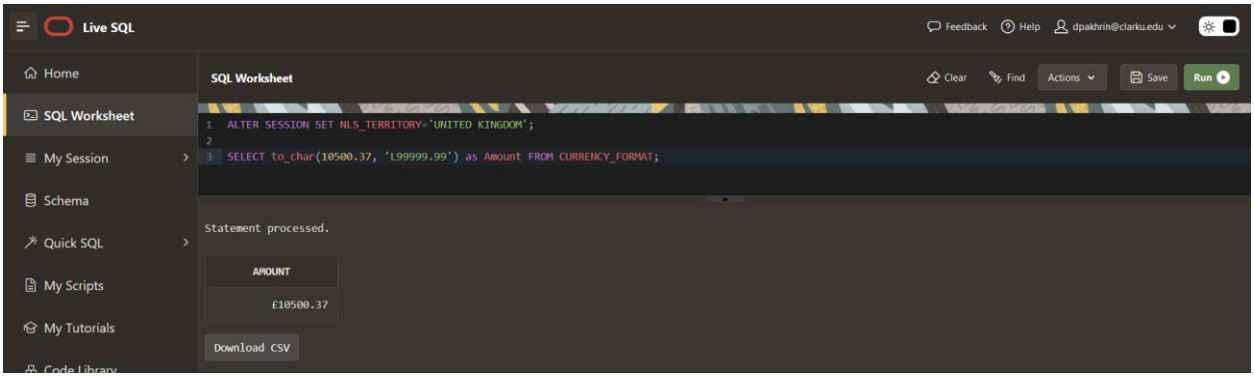
Here the operation inside the parenthesis is calculated first and the result is multiplied by 2 which gives an output of 16.

5. EXPLANATION
   In DBMS, there is the level of precedence of datatype. There may be an operation between an unidentical datatype and the operation makes the datatype of the result to the higher precedence. In other words, the lower precedence datatype is implicitly converted into a higher precedence datatype.
   For instance,
   SELECT 'AGE' || 24;

   Here we have two datatypes (AGE as VARCHAR and 24 as INT) which are concatenated by ||. Since VARCHAR has higher precedence, it implicitly converts INT into a String resulting an string output 'AGE24'.

6. SELECT (Screen Shot)





Section Two

7. SELECT (Screen Shot)

8.

a. SELECT (Screen Shot)

b. EXPLANATION

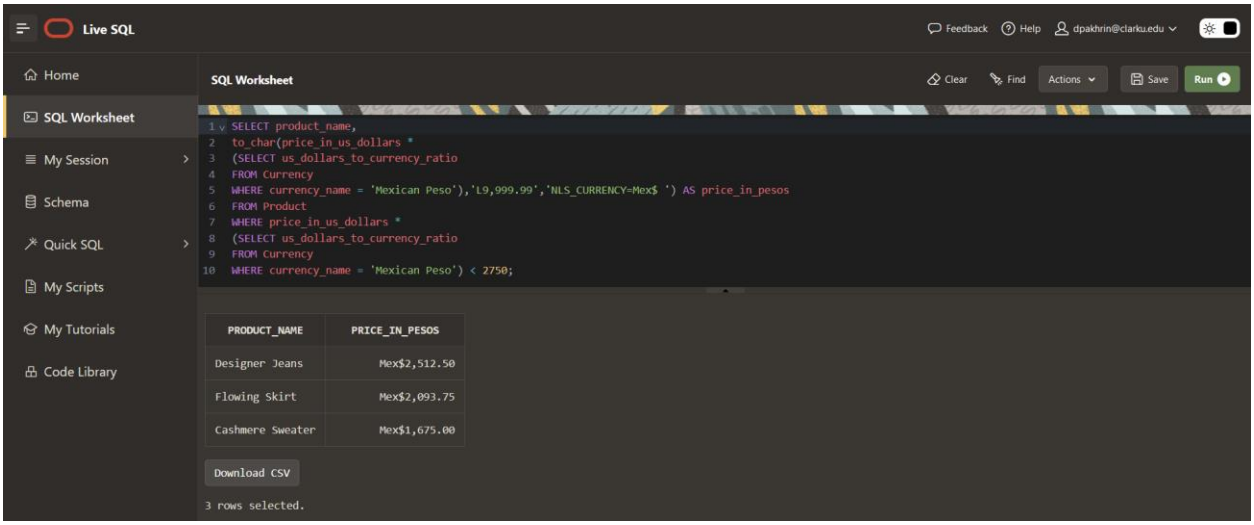
In the above query, we have three levels of subqueries. We want to display the price of the Flowing Skirt accepted in Cancun. For that we want the currency accepted in the Canun which is extracted from the Store location table. This is the most inner sub query which returns the currency id. The next sub query above the previous sub query will use the currency id to return the name of the currency. Finally, this name is used by the the last sub query to retrieve the US dollar to currency ratio. It is then used to convert the US dollar into a currency of our desire which is in our case a Mexican Dollar(Mex$). In this way, we can get our desired output in a single SQL query.

The advantage of using subquery:

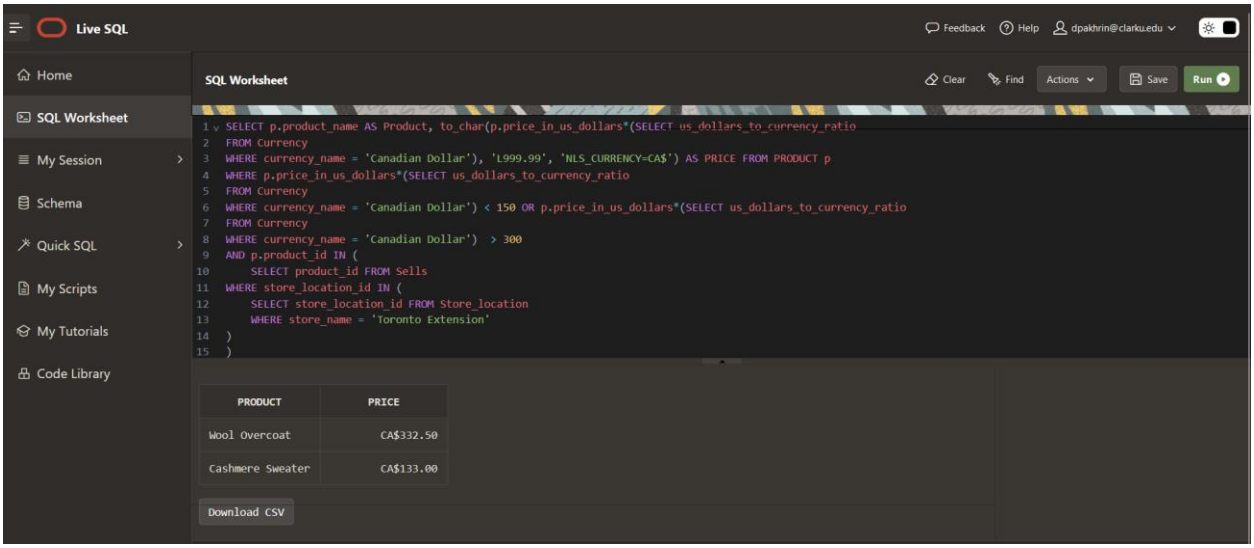
1) It has made the query more simple and readable

2) It has reduced the execution time of our query
3) It enables the code reusability

9. SELECT (Screen Shot)



10. SELECT (Screen Shot)

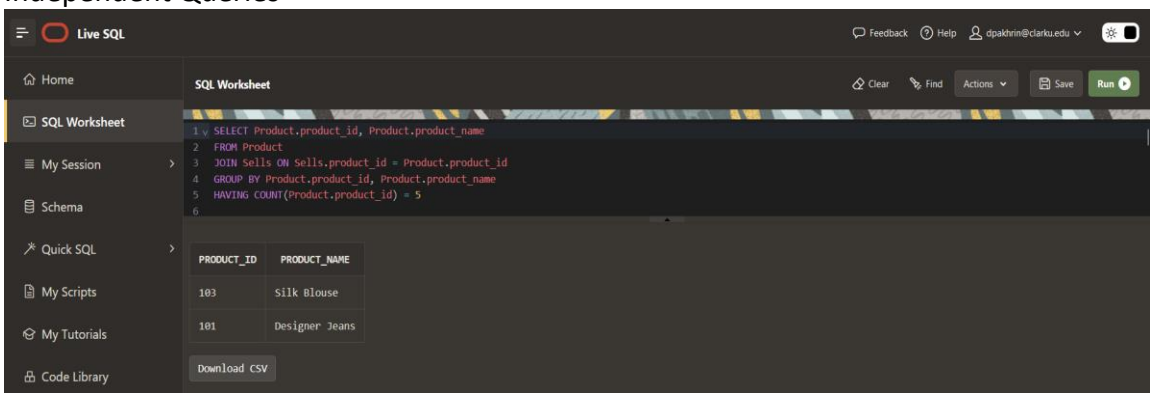


11.

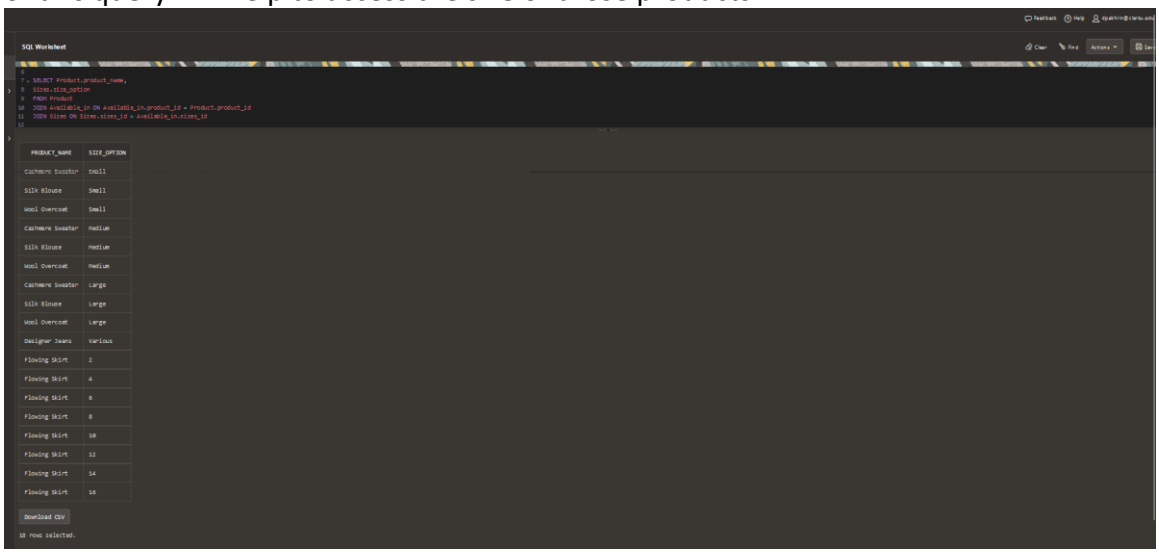
a. EXPLANATION

For the independent query, we can consider accessing the products id and product name that are available in every store location. For that we inner join the product table with the Sells table and group the rows by product id and product name which is available in every store(HAVING COUNT(Product.product_id) = 5). Next independent query can be accessing the product name and the size of the products that are available in every store. Since the relation between the Product table and Size table is shown by Available In table, we join this on the Product table.

b. Independent Queries

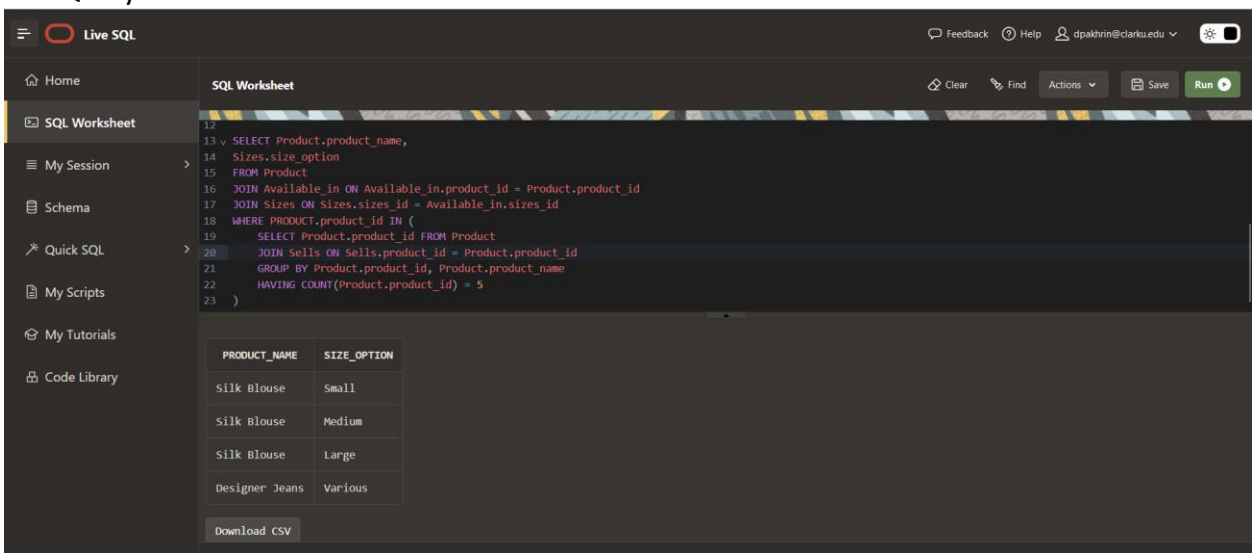


This query fetches all the products that are sold by every store. Later, the result of this query will help to access the size of those products.



This query fetches the sizes of all the available products. Later, this query is combined with another independent query to access our end result

c. Full Query

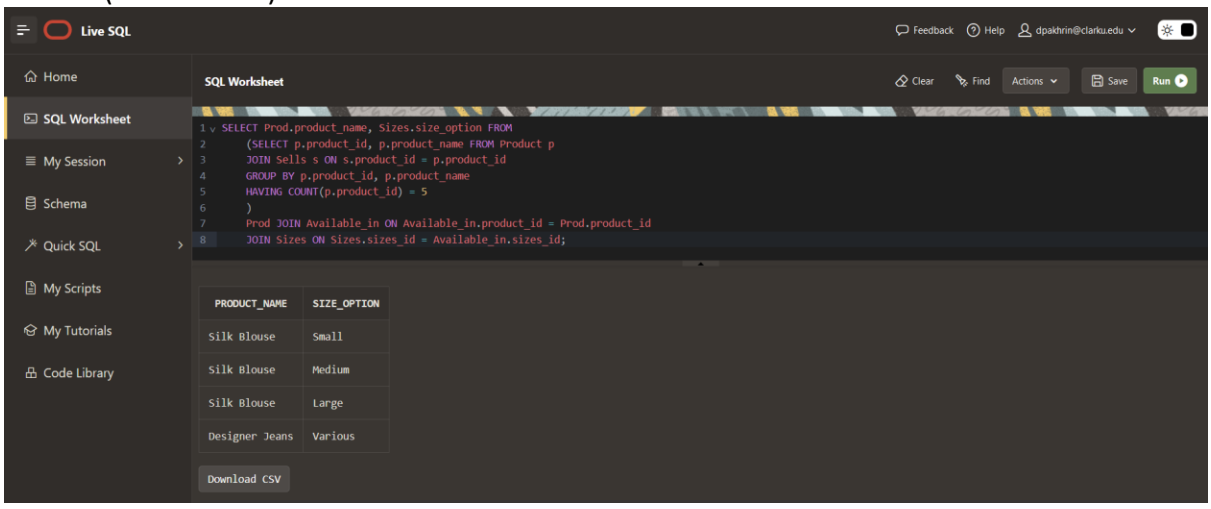


d. EXPLANATION

Finally, we have combined both independent queries in a single stand-alone query. In the first part, we have our second independent query which fetches the product name and the size of the query. Here, we have added a clause which is where clause. It is because we just want the sizes and the name of the product which is available in every store location. This is returned by our first independent clause. It is done by comparing with the product id which is sold by all the store locations.

12.

a. SELECT (Screen Shot)



b. EXPLANATION

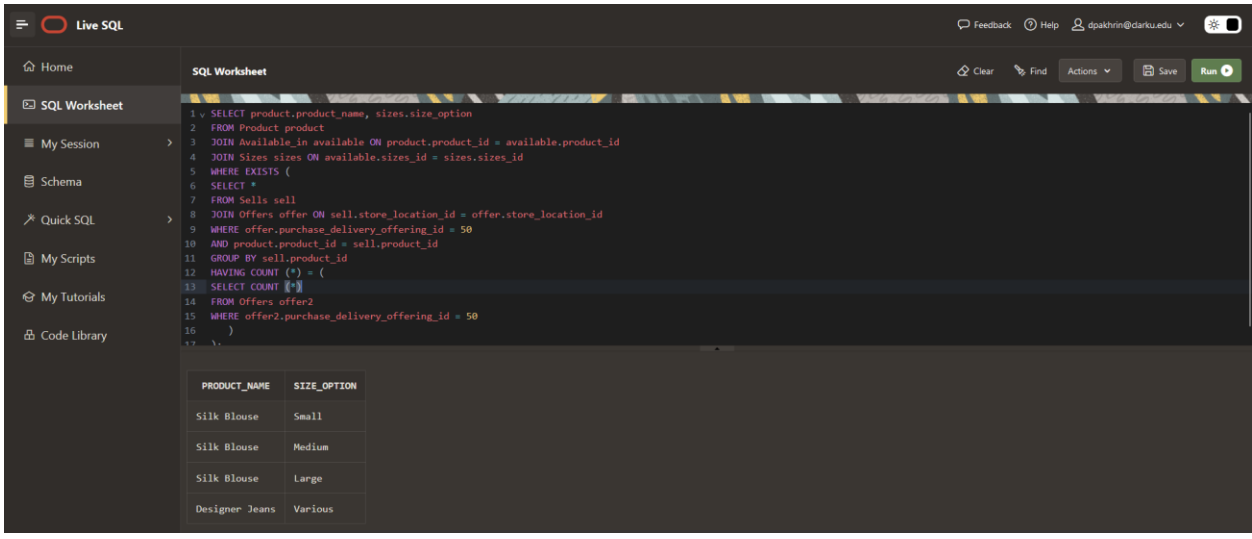
The subquery in the query mentioned above obtains the product name and product ID for each item that is offered at every store location. An alias that gives the result of the subquery a name is "Prod." This is the part of the join condition that merges the subquery result into the table that is already available. The FORM clause, instead of the WHERE clause, is where filtering is accomplished, as the subquery only returns products that meet Marcus's requirements.

13. Correlated Subquery vs Uncorrelated Subquery:
    a. Correlated Subquery
        - This subquery relies on the outer/main query so that the subquery gets executed at least once for every row in the main query.
        - It is joined with the Exists keyword.

    b. Uncorrelated Subquery

    This subquery is independent of the main query that can be executed without relying on the main query. It is only executed once and the output is passed to the main query.

14.
    a. SELECT (Screen Shot)



b. EXPLANATION
The above query is the another approach of obtaining the data that is similar to the problem 11. Here we used the EXISTS keyword where we define our sub query. It gets executed at least once for every row in the main query. Also we matched our the product_id of the outer query with the subquery (i.e) product.product_id = sell.product_id.

c. EXPLANATION
In my opinion, the query in problem 11 is more straightforward. The result obtained from the subquery is directly used by the outer query. In contrast, the query in this problem is complex and time-consuming as it gets executed at least once for every row in the main query.