ASSIGNMENT 1

Course- CS 561-A

Submitted By-

Yugaank Arun Sharma

Stevens Id- 10419077

Canvas Id- ysharma

In this assignment, you will write programs to evaluate relatively simple report queries and produce the output, and also express the queries in SQL. The key point of the exercise is to observe a large gap between the complexity of expressing the type of such queries and that of evaluating them. Your mission (in addition to writing the programs and SQL queries) is to consider the reasons for the gap (between the expression and evaluation of such queries) and how to narrow it.

To test my submission, follow these steps-

This is an eclipse project, so import the project in eclipse as FileSystem and import the folder present in the root folder of this submission by the name of 'ysharma'. Then change user name/password etc. that is needed to be done so you can test on your machine. And execute the program.

In the root folder along with the project folder, is another folder called 'SQL Queries' which contains two .sql files which generate the same output that the submitted java program does.

Differences that are between writing these queries and writing a program

1. In java, the program reads the database table row by row and can fetch those values that we require from each column. After the first row is read, the program logic becomes easier, It's only purpose is to add those values to the collection object or update in case there already is a value at the anticipated field. This is much easier than doing the same thing in SQL.
2. In SQL, you cannot just go on read one row at a time, we take the whole data that is required by a single query or we can use multiple queries to finally reach the destination. This is a lot more difficult than writing a program which can just do the same task by a single read of the table. Using multiple queries, when you cannot do a task in single query, is required and thus can be a bit more complicated.

3. But, what if the data is lot more than 500 rows. Well, we know java is preferred when It comes to garbage handling, but the complexity of the program would just become much more time consuming and hence will increase the complexity of both time and space. With SQL you can put on the same conditions, use joins, with clause, views (not so good as with clause in my opinion), or if you like sub queries, you can do that. Once you start understanding the logic that works behind the sql queries, the job can be much simpler and much less time consuming than writing a whole java or c++ program.
4. Sure, there is a huge gap between SQL and java, but if you are good on both, then why not just do it using SQL, which will save up time and space, and will be much more expressive.

SQL Queries

- **For 1st report-**

```
WITH
maxq as
(select B.prod as Product,B.quant as Max_Q,B.cust as CUSTOMER,B.day,B.month,B.year from (select prod , MAX(quant) as MAX_Q from sales group by prod) as A
        left join
        (select * from sales) as B
        on B.prod=A.prod and B.quant=A.MAX_Q),


minq as
(select D.prod as Product,D.quant as MIN_Q,D.cust as CUSTOMER,D.day,D.month,D.year from (select prod , MIN(quant) as MIN_Q from sales group by prod) as C
        left join
        (select * from sales) as D
        on D.prod=C.prod and D.quant=C.MIN_Q),

avgq as
        (select prod, AVG(quant) from sales group by prod)

select maxq.product,maxq.max_q,maxq.customer,concat(maxq.month,'/',maxq.day,'/',maxq.year) as MAX_DATE,minq.min_q,minq.customer,concat(minq.month,'/',minq.day,'/',minq.year) as MIN_DATE, avgq.avg from maxq join minq on maxq.product=minq.product join avgq on avgq.prod=maxq.product;
```

- **For 2nd report-**

WITH

CT1 as (select cust,prod,max(quant) as ctmax from sales A group by cust,prod,STATE),

CT2 as (select cust as CUSTOMER, prod as product, quant as quantity, day as ctday,month as ctmonth,year as ctyear from sales where state='CT' and year between 2000 and 2005),

NY1 as (select cust, prod, min(quant) as nymin from sales group by cust, prod,STATE),

NY2 as (select cust as CUSTOMER, prod as PRODUCT, quant as QUANTITY, day as nyday, month as nymonth, year as nyyear from sales where state='NY'),

NJ1 as (select cust, prod, min(quant) as njmin from sales group by cust, prod,STATE),

NJ2 as (select cust as CUSTOMER, prod as PRODUCT, quant as QUANTITY, day as njday, month as njmonth, year as njyear from sales where state='NJ'),

CT as (select foo1.CUSTOMER, foo1.product, foo1.quantity as CT_MAX, foo1.ctday, foo1.ctmonth, foo1.ctyear from (CT1 join CT2 on CT1.cust=CT2.CUSTOMER and CT1.prod=CT2.product and CT1.ctmax=CT2.quantity) as foo1),

NY as (select foo2.CUSTOMER, foo2.product, foo2.quantity as NY_MIN, foo2.nyday, foo2.nymonth, foo2.nyyear from (NY1 join NY2 on NY1.cust=NY2.CUSTOMER and NY1.prod=NY2.product and NY1.nymin=NY2.quantity) as foo2),

NJ as (select foo3.CUSTOMER, foo3.product, foo3.quantity as NJ_MIN, foo3.njday, foo3.njmonth, foo3.njyear from (NJ1 join NJ2 on NJ1.cust=NJ2.CUSTOMER and NJ1.prod=NJ2.product and NJ1.nJmin=NJ2.quantity) as foo3)

select * from ((CT natural full outer join NY) natural full outer join NJ) as result ORDER BY CUSTOMER

Both the SQL report outputs are same as java program outputs.