# Assignment 7 & 8

*Use the tables from Assignment 4.*

A. *Conjunctive selection operations can be written as a sequence of individual selections to generate minimal equivalent expressions. This further ensure the commutative property. So, write two forms of the same query, the non-optimized version and optimized version.*

1. Write a query to retrieve the products which have been generated from either 'Indonesia' or 'Brazil' and the price is less than 5 Rs.
2. Write a query to retrieve the customers whose first name starts with 'S' or 'R' and is of the gender 'F'.
3. Write a query to retrieve the count of products which have been generated from either 'Costa Rica' or 'Brazil' and is of the type 'Late' or 'Espresso'.
4. For each of the questions A.1-A.3, write the commutative version of the optimized queries.

B. *Cartesian Products and Theta joins. Write two queries where one is in the form of the cartesian product and the other, theta join.*

1. Write a query to retrieve the customer details who has ordered in the last one month.
2. Write a query to retrieve the customer IDs who has ordered the product originated from 'Brazil'.
3. Write a query to retrieve the First Name and Last Name of the customers who ordered product ID ='7'

*C. The next set of questions are based on rule based optimization which intends to reduce the cost in comparison before joining the intermediary tables. So, write two forms of the same query, the non-optimized version and optimized version.*

1. Write a query to retrieve the details of the products which are 'Espresso' and has customerID of 5

2. Write a query to retrieve the details of the customers who are ‘F’ and ordered ‘Latte’.
3. Write a query to retrieve the customer IDs who have ordered a product which originated from ‘Brazil’ and has been ordered at least twice by the customer.
4. Write a query to retrieve the number of ‘M’ customers who have ordered a product in September, 2020.

D. *Subquery using IN or Inner Join. Write two forms of the same query, the non-optimized version (using ‘IN’ in sub-query) and optimized version (using ‘inner join’)*

1. Write a query to find the list of productIDs which were ordered by customers with last name is Taylor or Bluth or Armstrong.
2. Write a query to find the date of order for products that were ordered by customers whose first name is Katie or John or George.
3. Write a query to find out the name of all the customerIDs who ordered coffee that originated from ‘Costa Rica’ or ‘Indonesia’.
4. Write a query to find out the first name and last name of all the customers who ordered products with ID as either 1 or 3 or 5 or 7.

E. *Join operations are associative and commutative. The optimized version must join those two tables first which yield less number of entries, and then apply the other join. This also applies for joining two tables. Therefore, write the optimized version of the joining*

1. Write a query to find the list of customerID, first name, last name and the origin of the products that they ordered in September 2020.
2. Write a query to find out the newest to oldest orders placed by the customer whose last name is ‘Martin’. List the name of the product, price, first name, last name, and order date and time.

F. *Distribution of Selection Operations during join operation. Applying a selection after doing the Theta Join causes all the tuples returned by the Theta Join to be monitored after the join. Therefore, the optimized query ensures applying first selection to the table that results in a fewer number of tuples, and then join it with the other table.*

1. Write a query to find out the order ID, first name, last name, contact number, and order time of top 10 oldest to newest orders.
2. Write a query to find out the product ID, first name, last name, customer ID, and order time of top 10 oldest to newest orders.
3. Write a query to retrieve all the details of the customers who have never ordered a product.

References :

1. https://www.geeksforgeeks.org/query-optimization-in-relational-algebra/
2. http://www.cbcb.umd.edu/confcour/Spring2014/CMSC424/query_optimization.pdf
3. https://www.tutorialcup.com/dbms/query-optimization.htm
4. https://www.red-gate.com/simple-talk/sql/learn-sql-server/using-covering-indexes-to-improve-query-performance/
5. https://www.oreilly.com/library/view/high-performance-mysql/9780596101718/ch04.html

Examples :

1. *Conjunctive Selection Operation*

Q. Write a query to retrieve the products which have been generated from ‘Costa Rica’ and the price is greater than 10 Rs

Ans :

*Non-optimized Version*

```
select * from product where CoffeeOrigin=’Costa Rica’
```

intersect

select * from product where Price >10

*Optimized Version*

select * from product where ProductID in (select ProductID from product where CoffeeOrigin='Costa Rica) and Price >10

The optimized version is commutative in nature, we can replace the inner and outer selection respectively. In practice, it is better and more optimal to apply that selection first which yields a fewer number of tuples.

2. Write a query to retrieve the customer IDs who has ordered the product originated from 'Brazil'.

A :
*Non-optimized Version*

select order.customerID from product cross join orders where product.coffeeOrigin = 'Brazil' and product.customerID=order.customerID

*Optimized Version*

select order.customerID from product inner join *orders* ON product.customerID=order.customerID where product.coffeeOrigin = 'Brazil'

3. Write a query to retrieve the details of the products which are 'Espresso' and price is greater than 5 Rs

*A: Non-optimized Version*

select * from product pd, orders od where pd.productID = od.productID

and pd.Name='Espresso' and od.CustomerID='5'

*Optimized Version*

```
 select * from
(select * from product where Name='Espresso') pd
(select * from orders where CustomerID='5') od
where pd.productID = od.productID
```