**EECS 116 :**

**Homework 7**

**1. [10 pts] Create an SQL query that finds customer id, customer email, flight\_number,**

**projected\_departure\_datetime, and the quantity of tickets for each reservation made by**

**customers with an email starting with letter ‘ i’ .**

SELECT cid, email, flight\_number, projected\_departure\_datetime, sum(quantity)

FROM cs122a.customer natural join customer\_reserves\_flight

where email like 'i%'

group by ssn;

**2. [10 pts] Create the same query to return the same attributes. The only difference is that the filtering condition is “email ending with d.com ” instead of “starting with letter i ”.**

SELECT C.cid, email, flight\_number, projected\_departure\_datetime, sum(quantity)

FROM cs122a.customer C left join customer\_reserves\_flight CC on C.cid=CC.cid

where email like '%d.com'

group by ssn;

**3. [10 pts] Create the same query as question 1 to return the same attributes. The only**

**change that you need to make is the filtering condition is the**

**projected\_departure\_datetime is on or after ‘20151001**

**00:00:00’.**

SELECT cid, email, flight\_number, projected\_departure\_datetime, sum(quantity)

FROM cs122a.customer natural join customer\_reserves\_flight

where projected\_departure\_datetime <= '2015-07-01'

group by ssn;

**4. We want to create indexes to speed up three queries above. To make the right**

**decision on the indexes, first check the visual query plan for each query above by copying and pasting your SQLs and clicking the lightning + magnifier button in MySQLWorkBench like the following example. You are not required to submit the visual query plan. Now, submit the textual query plan for each of the three queries. To create a textual query plan, you can use the command “EXPLAIN” in front of each query to see the query plan in text (e.g., “EXPLAIN SELECT email from Customer …”). After adding “EXPLAIN” to a query, click “Query” > “Execute (All or Selection) to Text” to generate the output andpaste the output as an answer.**

**a. Query 1)**

-> explain SELECT cid, email, flight\_number, projected\_departure\_datetime, sum(quantity)

FROM cs122a.customer natural join customer\_reserves\_flight

where email like 'i%'

group by ssn;

| 1 | SIMPLE | customer\_reserves\_flight | | ALL | PRIMARY | | | | 25 | 100.00 | Using temporary; Using filesort |

| 1 | SIMPLE | customer | | ALL | PRIMARY | | | | 20 | 5.00 | Using where; Using join buffer (Block Nested Loop) |

+ ------- + ---------------- + ---------- + --------------- + --------- + ------------------ + -------- + ------------ + -------- + --------- + ------------- + ---------- +

2 rows

**b. Query 2)**

Execute:

> explain SELECT C.cid, email, flight\_number, projected\_departure\_datetime, sum(quantity)

FROM cs122a.customer C left join customer\_reserves\_flight CC on C.cid=CC.cid

where email like '%d.com'

group by ssn

+ ------- + ---------------- + ---------- + --------------- + --------- + ------------------ + -------- + ------------ + -------- + --------- + ------------- + ---------- +

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+ ------- + ---------------- + ---------- + --------------- + --------- + ------------------ + -------- + ------------ + -------- + --------- + ------------- + ---------- +

| 1 | SIMPLE | C | | ALL | | | | | 20 | 11.11 | Using where; Using temporary; Using filesort |

| 1 | SIMPLE | CC | | ALL | PRIMARY | | | | 25 | 100.00 | Using where; Using join buffer (Block Nested Loop) |

+ ------- + ---------------- + ---------- + --------------- + --------- + ------------------ + -------- + ------------ + -------- + --------- + ------------- + ---------- +

2 rows

**c. Query 3)**

Execute:

> explain SELECT cid, email, flight\_number, projected\_departure\_datetime, sum(quantity)

FROM cs122a.customer natural join customer\_reserves\_flight

where projected\_departure\_datetime <= '2015-07-01'

group by ssn

+ ------- + ---------------- + ---------- + --------------- + --------- + ------------------ + -------- + ------------ + -------- + --------- + ------------- + ---------- +

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+ ------- + ---------------- + ---------- + --------------- + --------- + ------------------ + -------- + ------------ + -------- + --------- + ------------- + ---------- +

| 1 | SIMPLE | customer\_reserves\_flight | | ALL | PRIMARY | | | | 25 | 33.33 | Using where; Using temporary; Using filesort |

| 1 | SIMPLE | customer | | eq\_ref | PRIMARY | PRIMARY | 4 | cs122a.customer\_reserves\_flight.cid | 1 | 100.00 | |

+ ------- + ---------------- + ---------- + --------------- + --------- + ------------------ + -------- + ------------ + -------- + --------- + ------------- + ---------- +

2 rows

**5. [10 pts] Write and execute a “CREATE INDEX” statement that creates an index on the Customer.email attribute. The index name should be “ix\_Customer\_email”.**

Create index ix\_Customer\_email

on customer (email asc)

**6. [10 pts] Write and execute a “CREATE INDEX” statement that creates an index on the Customers\_Reserves\_Flight.projected\_departure\_datetime attribute. The index name should be “ix\_CRF\_projected\_departure\_datetime”.**

Create index ix\_CRF\_projected\_departure\_datetime

on customer\_reserves\_flight (projected\_departure\_datetime asc)

**7. [15 pts] Now you have created two indexes. Check the visual query plan again for each of the three queries. Again, you are not required to submit the visual query plans. However, you are required to submit the textual “EXPLAIN” output of your queries by following the same steps in Q4.**

**a. Query 1)**

Execute:

> explain SELECT cid, email, flight\_number, projected\_departure\_datetime, sum(quantity)

FROM cs122a.customer natural join customer\_reserves\_flight

where email like 'i%'

group by ssn

+ ------- + ---------------- + ---------- + --------------- + --------- + ------------------ + -------- + ------------ + -------- + --------- + ------------- + ---------- +

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+ ------- + ---------------- + ---------- + --------------- + --------- + ------------------ + -------- + ------------ + -------- + --------- + ------------- + ---------- +

| 1 | SIMPLE | customer | | range | PRIMARY,ix\_Customer\_email | ix\_Customer\_email | 93 | | 3 | 100.00 | Using index condition; Using temporary; Using filesort |

| 1 | SIMPLE | customer\_reserves\_flight | | ref | PRIMARY | PRIMARY | 4 | cs122a.customer.cid | 1 | 100.00 | |

+ ------- + ---------------- + ---------- + --------------- + --------- + ------------------ + -------- + ------------ + -------- + --------- + ------------- + ---------- +

2 rows

**b. Query 2)**

Execute:

> explain SELECT C.cid, email, flight\_number, projected\_departure\_datetime, sum(quantity)

FROM cs122a.customer C left join customer\_reserves\_flight CC on C.cid=CC.cid

where email like '%d.com'

group by ssn

+ ------- + ---------------- + ---------- + --------------- + --------- + ------------------ + -------- + ------------ + -------- + --------- + ------------- + ---------- +

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+ ------- + ---------------- + ---------- + --------------- + --------- + ------------------ + -------- + ------------ + -------- + --------- + ------------- + ---------- +

| 1 | SIMPLE | C | | ALL | | | | | 20 | 11.11 | Using where; Using temporary; Using filesort |

| 1 | SIMPLE | CC | | ref | PRIMARY | PRIMARY | 4 | cs122a.C.cid | 1 | 100.00 | |

+ ------- + ---------------- + ---------- + --------------- + --------- + ------------------ + -------- + ------------ + -------- + --------- + ------------- + ---------- +

2 rows

**c. Query 3)**

Execute:

> explain SELECT cid, email, flight\_number, projected\_departure\_datetime, sum(quantity)

FROM cs122a.customer natural join customer\_reserves\_flight

where projected\_departure\_datetime <= '2015-07-01'

group by ssn

+ ------- + ---------------- + ---------- + --------------- + --------- + ------------------ + -------- + ------------ + -------- + --------- + ------------- + ---------- +

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+ ------- + ---------------- + ---------- + --------------- + --------- + ------------------ + -------- + ------------ + -------- + --------- + ------------- + ---------- +

| 1 | SIMPLE | customer\_reserves\_flight | | range | PRIMARY,ix\_CRF\_projected\_departure\_datetime” | ix\_CRF\_projected\_departure\_datetime” | 5 | | 2 | 100.00 | Using index condition; Using temporary; Using filesort |

| 1 | SIMPLE | customer | | eq\_ref | PRIMARY | PRIMARY | 4 | cs122a.customer\_reserves\_flight.cid | 1 | 100.00 | |

+ ------- + ---------------- + ---------- + --------------- + --------- + ------------------ + -------- + ------------ + -------- + --------- + ------------- + ---------- +

2 rows

**8. [10 pts] Now, by comparing the query plan of Q4 and Q7, briefly explain how the indexes are being used by the query optimizer to make these queries run faster.**

The queries we used have the ‘where’ clause as the conditioning statement. After we create indexes, the query optimizer is able to determine quickly which rows matches the condition instead of going through the entire table.

**9. [10 pts] If you carefully examine the execution plan of Query 1 and Query 2 after creating the ix\_Customer\_Email index, you can notice that the index is being used for Query 2. Can you think of a reason why this index is being used?**

We are using the where clause except when we index the right values the lookup is immediate and lookup is faster. The query optimizer chooses full index scan not full table scan as all the requested information is fully contained in the index, no access to the data table will be required.

**[Extra Point Questions]**

1. [10 pts] Consider the following SQL query.

SELECT total\_amount, count(\*) from

DishOrder DO, Dish D, Lounge L

WHERE DO.lid=D.lid and DO.total\_aD.price > 30 and DO.lid = L.lid and L.airport\_IATA\_code like 'S%'

GROUP BY DO.total\_amount

HAVING count(\*) > 1

ORDER BY DO.total\_amount;

First, check its visual query plan (no need to turn it in). Then find a way to improve its

execution speed by creating index(es). To support your argument, attach the

“EXPLAIN” output of the query after you create the index(es).

a. The textual “EXPLAIN” plan of the original query.

Execute:

> explain SELECT total\_amount, count(\*) from DishOrder DO, Dish D, Lounge L

WHERE DO.lid=D.lid and DO.total\_amount > 300 and

D.price > 30 and DO.lid = L.lid and L.airport\_IATA\_code like 'S%'

GROUP BY DO.total\_amount

HAVING count(\*) > 1

ORDER BY DO.total\_amount

+ ------- + ---------------- + ---------- + --------------- + --------- + ------------------ + -------- + ------------ + -------- + --------- + ------------- + ---------- +

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+ ------- + ---------------- + ---------- + --------------- + --------- + ------------------ + -------- + ------------ + -------- + --------- + ------------- + ---------- +

| 1 | SIMPLE | DO | | ALL | lid | | | | 18 | 33.33 | Using where; Using temporary; Using filesort |

| 1 | SIMPLE | L | | eq\_ref | PRIMARY,airport\_IATA\_code | PRIMARY | 4 | cs122a.DO.lid | 1 | 69.23 | Using where |

| 1 | SIMPLE | D | | ref | PRIMARY | PRIMARY | 4 | cs122a.DO.lid | 2 | 33.33 | Using where |

+ ------- + ---------------- + ---------- + --------------- + --------- + ------------------ + -------- + ------------ + -------- + --------- + ------------- + ---------- +

3 rows

b. Your statement(s) to create index(es).

CREATE INDEX ix\_DishOrder\_total\_amount

ON DishOrder (total\_amount);

c. The textual “EXPLAIN” plan of the query after creating the index(es).

Execute:

> explain SELECT total\_amount, count(\*)

from DishOrder DO, Dish D, Lounge L

WHERE DO.lid=D.lid and DO.total\_amount > 300 and

D.price > 30 and DO.lid = L.lid and L.airport\_IATA\_code like 'S%'

GROUP BY DO.total\_amount

HAVING count(\*) > 1

ORDER BY DO.total\_amount

+ ------- + ---------------- + ---------- + --------------- + --------- + ------------------ + -------- + ------------ + -------- + --------- + ------------- + ---------- +

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+ ------- + ---------------- + ---------- + --------------- + --------- + ------------------ + -------- + ------------ + -------- + --------- + ------------- + ---------- +

| 1 | SIMPLE | DO | | range | lid,ix\_dishorder\_ok,ix\_DishOrder\_total\_amount | ix\_DishOrder\_total\_amount | 5 | | 3 | 100.00 | Using index condition; Using where |

| 1 | SIMPLE | L | | eq\_ref | PRIMARY,airport\_IATA\_code,ix\_lounge\_ok | PRIMARY | 4 | cs122a.DO.lid | 1 | 69.23 | Using where |

| 1 | SIMPLE | D | | ref | PRIMARY,ix\_dish\_ok | PRIMARY | 4 | cs122a.DO.lid | 2 | 33.33 | Using where |

+ ------- + ---------------- + ---------- + --------------- + --------- + ------------------ + -------- + ------------ + -------- + --------- + ------------- + ---------- +

3 rows