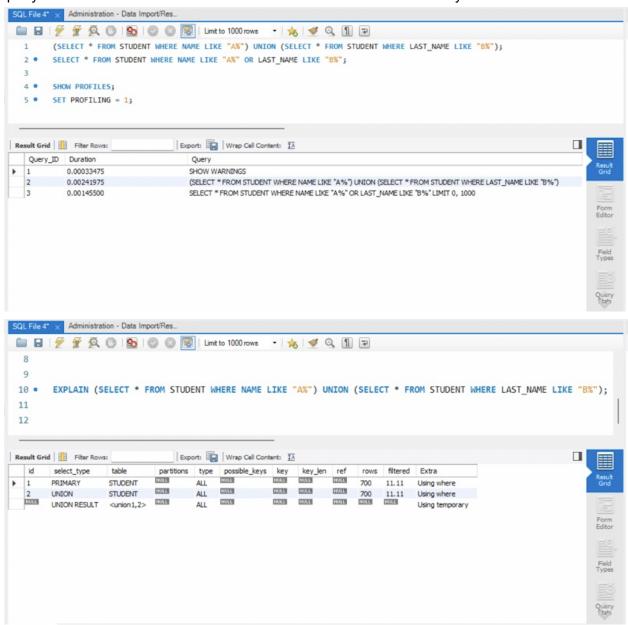
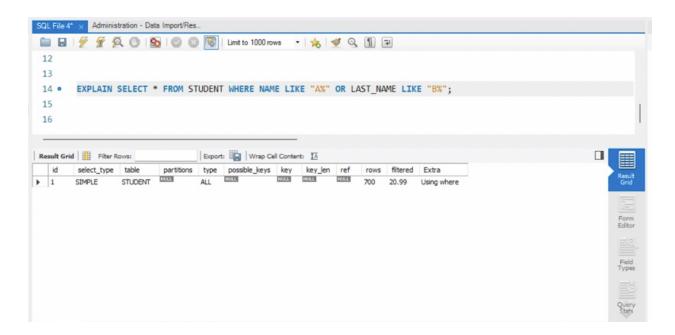
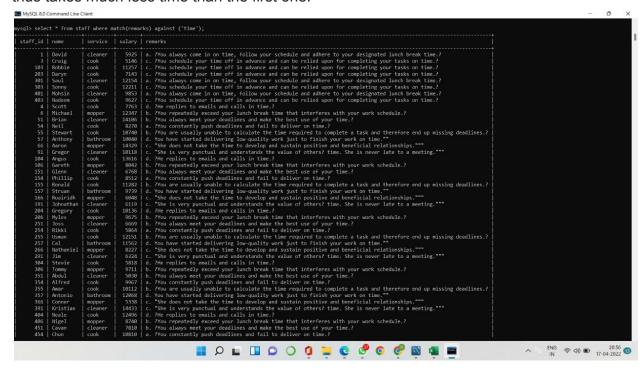
1. The query (having Like statement) for searching in the text-type column with the clause *UNION* is optimized using the *OR* clause.

We infer from the following figure that query with *OR* clause takes *0.001455* units which is less than that of query with union clause (*0.00241975* units). This is because the first query scans each record in the table twice while the second scans only once.





select \* from staff where remarks like '%time%';
 select \* from staff where match(remarks) against ('time');
 Unlike the first query, the second one uses a full text index against the column and thus takes much less time than the first one.



```
mysql> explain select * from staff where match(remarks) against ('time');

| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |

| 1 | SIMPLE | staff | NULL | fulltext | remarks_ft | remarks_ft | 0 | const | 1 | 100.00 | Using where; Ft_hints: sorted |

1 row in set, 1 warning (0.00 sec)

mysql> explain select * from staff where remarks like '%time%';

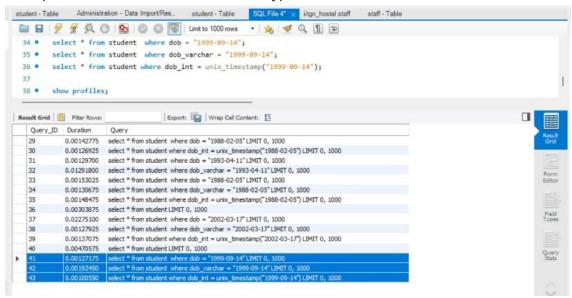
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |

1 | SIMPLE | staff | NULL | ALL | NULL | NULL | NULL | 499 | 11.11 | Using where |

1 row in set, 1 warning (0.00 sec)
```

3. Modified the roll\_no datatype from 'INT' to 'TINYINT'. This decreased the execution time. This is because smaller data types are usually faster, because they use less space on the disk, in memory, and in the CPU cache. They also generally require fewer CPU cycles to process. Same reason can be applied to mediumint

4. We altered the table by adding two new columns dob\_varchar(with dob as varchar data type) and dob\_int (with dob as int data type)) to the students table. The values have to be taken from the dob(stored as date data type) column. We infer from the following that int comparisons are faster, for the simple fact that ints take up much less space than varchar and date data types.



5. Searching with isnull works faster than without explicitly writing not null.

```
mysql> SELECT COUNT(SALARY) FROM STAFF;
 COUNT(SALARY)
           443
 row in set (0.00 sec)
mysql> SELECT COUNT(SALARY) FROM STAFF WHERE SALARY IS NOT NULL;
 COUNT(SALARY)
           443
 row in set (0.00 sec)
mysql> SHOW PROFILES;
 Query_ID | Duration
                       Query
       17 | 0.37439150 | ALTER TABLE TRANSACTION MODIFY TRANSACTION_ID MEDIUMINT
       18
           0.26193000 | ALTER TABLE TRANSACTION MODIFY TRANSACTION_ID MEDIUMINT()
       19 | 0.00013850 | SET PROFILING
       20 | 0.06792775 | SET PROFILING=1
       21 | 0.28855750 | SELECT * FROM STUDENT LIMIT 5
          0.02267700
       22
                         SELECT ROLL NO FROM STUDENT HOSTEL PAYMENT
       23
          | 0.03129800 | SELECT DATABASE()
       24 | 0.02922075 | SELECT ROLL NO FROM STUDENT HOSTEL PAYMENT
       25 | 0.19772825 |
                        SELECT COUNT(*) FROM STAFF
            0.00070025
       26
                         SELECT COUNT(SALARY) FROM STAFF
       27 | 0.01663100 | SELECT COUNT(SALARY) FROM STAFF WHERE SALARY NOT NULL
       28 | 0.00062875 |
                        SELECT COUNT(SALARY) FROM STAFF WHERE SALARY IS NOT NULL
                         UPDATE STAFF SET SALARY = NULL WHERE SALARY < 6000
       29 | 0.07104075 |
           0.00080000
                         SELECT COUNT(SALARY) FROM STAFF
       30
       31 | 0.00063450 | SELECT COUNT(SALARY) FROM STAFF WHERE SALARY IS NOT NULL
l5 rows in set, 1 warning (0.00 sec)
```

- 6. mysql caches the select query as well as the result set, allowing identical selects to run faster as data is fetched from the memory cache. We are not able to perform caching because the mysql version 8.0 does not support it.
- 7. We performed *JOIN* on warden and furniture tables and compared it with a subquery. We infer that retrieval time of the query using joins is faster than that of a subquery thus helps better in the optimization. This is because the subquery runs the internal queries first and then from the resulting table again filters out the actual results while *JOIN*

produces the result in one go.



id   select_type	table	partitions		possible_keys	key	key_len				
1   SIMPLE	warden	NULL	ref	block_name_idx	block_name_idx	83	const	49		Using where
1   SIMPLE	<pre><subquery2></subquery2></pre>	NULL	eq_ref	<pre><auto_distinct_key></auto_distinct_key></pre>	<auto_distinct_key></auto_distinct_key>	83	final.warden.block_name	1	100.00	Using where
2   MATERIALIZED	furniture	NULL	ref	block_name	block_name	83	const	49	100.00	Using index



However, more joins in a query means the database server has to do more work, which means that it is a more time consuming process to retrieve data.





8. The execution time is decreased as we used natural join instead of subqueries. This is because Subquery runs the internal queries first and then from the result set again filters out the actual results. Wherein join runs the and produces the result in one go.

rsql> show profiles;
Query_ID   Duration   Query
19   0.42159175   select name from student where roll_no in (select roll_no from postgrad where block_name in (select block_name from furniture where room_no in (select room_no from hostel_rooms where sharing_type ='Double')))   20   0.01484350   select student.name from student natural join postgrad natural join furniture natural join hostel_rooms where sharing_type='Double'
rows in set, 1 warning (0.00 sec)

select filtered	Extra				possible_keys 	key	key_len	
								+
SIMPLE 100.00		postgrad where; Using			PRIMARY,block_name_idx	block_name_idx	83	NULL
SIMPLE 100.00		student	NULL	eq_ref	PRIMARY	PRIMARY		final.postgrad.roll_no
SIMPLE		hostel_rooms where; Using			PRIMARY	NULL	NULL	NULL
SIMPLE		furniture index: End te	NULL		block_name	block_name	88	final.postgrad.block_name,final.hostel_rooms.room_no

d   select_type   iltered   Extra		partitions 		possible_keys	key	key_len	ref	row
4   5711015								
1   SIMPLE   100.00   Using wh	furniture pere: Using ind	NULL dex i	Index	block_name	block_name	88	NULL	38
	hostel_rooms		eq_ref	PRIMARY,block_name	PRIMARY	86	final.furniture.room_no,final.furniture.block_name	
1   SIMPLE   100.00   Using in	postgrad ndex	NULL	ref	PRIMARY,block_name_idx	block_name_idx	83	final.furniture.block_name	
1   SIMPLE   100.00   NULL	student	NULL	eq_ref	PRIMARY	PRIMARY	4	final.postgrad.roll_no	

## Group G1 Contribution (out of 50%)

-----

Shubham Ghosh 21120017- 12.5% Pranav Ninawe 21120015 - 12.5% Rovin Singh 21120016 - 12.5% Darshan Patil 21120006 - 12.5% (Equal participation of all group members)

## Group G2 Contribution (out of 50%)

\_\_\_\_\_

Iram Nawab 21250011 - 12.5% Manasvi Kothari 21120011 - 12.5% Hardik Mahur 19110086 - 12.5% Shoaib Alam 21250024 - 12.5% (Equal participation of all group members)