Question-1:-

Table: Passenger

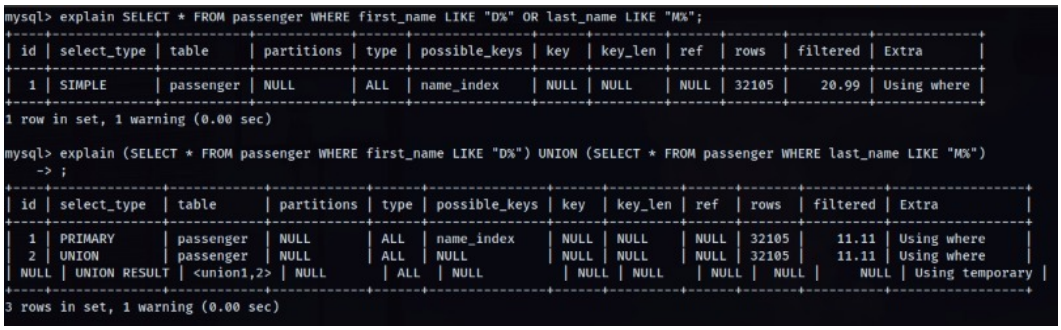
Attributes: first_name, last_name

Justification: We have a table passenger where we are doing text search on two columns First name and Last name. We found optimization when we used Union on two subqueries instead of Or in a query.

When we use Or for search in two columns, MySql uses the index based seeks only on one of the two columns and does a linear search on the other. But when we use Union for two queries, it is forced to used the index based seeks for both the columns taht results in reducing the overall time for execution.

Query_ID	Duration	Query
1	0.04327750	SELECT * FROM passenger WHERE first_name LIKE 'd%' OR last_name LIKE 'm%'
2	0.03628350	(SELECT * FROM passenger WHERE first_name LIKE 'd%') UNION (SELECT * FROM passenger WHERE last_name LIKE 'm%')

This Image shows decrease in time after using Union on two subqueries istead of Or.



```
mysql> explain SELECT * FROM passenger WHERE first_name LIKE "D%" OR last_name LIKE "M%";
+----+-------------+-----------+------------+------+---------------+------------+---------+------+-------+----------+-------------+
| id | select_type | table     | partitions | type | possible_keys | key        | key_len | ref  | rows  | filtered | Extra       |
+----+-------------+-----------+------------+------+---------------+------------+---------+------+-------+----------+-------------+
|  1 | SIMPLE      | passenger | NULL       | ALL  | name_index    | NULL       | NULL    | NULL | 32105 |    20.99 | Using where |
+----+-------------+-----------+------------+------+---------------+------------+---------+------+-------+----------+-------------+
1 row in set, 1 warning (0.00 sec)

mysql> explain (SELECT * FROM passenger WHERE first_name LIKE "D%") UNION (SELECT * FROM passenger WHERE last_name LIKE "M%")
    -> ;
+------+--------------+------------+------------+------+---------------+------+---------+------+-------+----------+-----------------+
| id   | select_type  | table      | partitions | type | possible_keys | key  | key_len | ref  | rows  | filtered | Extra           |
+------+--------------+------------+------------+------+---------------+------+---------+------+-------+----------+-----------------+
|    1 | PRIMARY      | passenger  | NULL       | ALL  | name_index    | NULL | NULL    | NULL | 32105 |    11.11 | Using where     |
|    2 | UNION        | passenger  | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 32105 |    11.11 | Using where     |
| NULL | UNION RESULT | <union1,2> | NULL       | ALL  | NULL          | NULL | NULL    | NULL |  NULL |     NULL | Using temporary |
+------+--------------+------------+------------+------+---------------+------+---------+------+-------+----------+-----------------+
3 rows in set, 1 warning (0.00 sec)
```

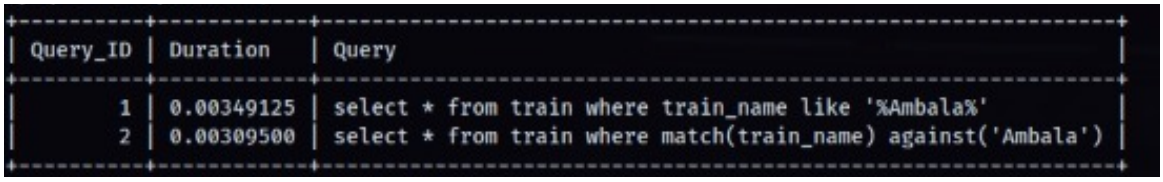
This image shows the number of row scans for the Union and Or sql statements run for this question.

Question-2:-

Table: Train

Attributes: train_name

Justification: We have initially searched for a specific text pattern on unoptimized table. After that, we have applied fulltext indexing on the column train_name. This resulted in lesser number of scans when searched using indexing and also decreasing the time of execution.



Query_ID	Duration	Query
1	0.00349125	select * from train where train_name like '%Ambala%'
2	0.00309500	select * from train where match(train_name) against('Ambala')

This image shows the decrease in the execution time after indexing the column Train_Name.

```
mysql> explain SELECT train_id, train_name FROM train WHERE train_name LIKE '%Ambala%';
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra       |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------------+
|  1 | SIMPLE      | train | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 3964 |    11.11 | Using where |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)

mysql> explain SELECT train_id, train_name FROM train WHERE match(train_name) AGAINST ('Ambala');
+----+-------------+-------+------------+----------+---------------+-------+---------+-------+------+----------+-------------------------------+
| id | select_type | table | partitions | type     | possible_keys | key   | key_len | ref   | rows | filtered | Extra                         |
+----+-------------+-------+------------+----------+---------------+-------+---------+-------+------+----------+-------------------------------+
|  1 | SIMPLE      | train | NULL       | fulltext | ft_tn         | ft_tn | 0       | const |    1 |   100.00 | Using where; Ft_hints: sorted |
+----+-------------+-------+------------+----------+---------------+-------+---------+-------+------+----------+-------------------------------+
1 row in set, 1 warning (0.01 sec)
```

This image shows the number of rows scanned before and after indexing(optimizing).

Question-3:-

Table: Stall

Attributes: stall_id

Justification: We have lesser number of tuples in this table. So, we have the option of changing the data type of Stall_id from int to TINYINT. So, we have examined the select query for searching for a specific Stall_id before and after altering the data type. We found out that this TINYINT data structure is giving an optimized search. This is because, the range of values in TINYINT are much less less than that of Integer values. So, the search is narrowed down in a way.

```
+----------+-------------+-----------------------------------------------+
| Query_ID | Duration    | Query                                         |
+----------+-------------+-----------------------------------------------+
|        1 | 0.00426400  | select * from stall where stall_id=76         |
|        2 | 0.01046375  | alter table stall modify column stall_id tinyint |
|        3 | 0.11179600  | alter table stall modify column stall_id tinyint |
|        4 | 0.00115750  | select * from stall where stall_id=76         |
+----------+-------------+-----------------------------------------------+
```

This image show the decrease in time after changing the data type.

```
Database changed
mysql> explain select * from stall where stall_id=76;
+----+-------------+-------+------------+-------+---------------+---------+---------+-------+------+----------+-------+
| id | select_type | table | partitions | type  | possible_keys | key     | key_len | ref   | rows | filtered | Extra |
+----+-------------+-------+------------+-------+---------------+---------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | stall | NULL       | const | PRIMARY       | PRIMARY | 4       | const |    1 |   100.00 | NULL  |
+----+-------------+-------+------------+-------+---------------+---------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)

mysql> alter table stall modify column stall_id tinyint;
Query OK, 114 rows affected (0.11 sec)
Records: 114  Duplicates: 0  Warnings: 0

mysql> explain select * from stall where stall_id=76;
+----+-------------+-------+------------+-------+---------------+---------+---------+-------+------+----------+-------+
| id | select_type | table | partitions | type  | possible_keys | key     | key_len | ref   | rows | filtered | Extra |
+----+-------------+-------+------------+-------+---------------+---------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | stall | NULL       | const | PRIMARY       | PRIMARY | 1       | const |    1 |   100.00 | NULL  |
+----+-------------+-------+------------+-------+---------------+---------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)
```

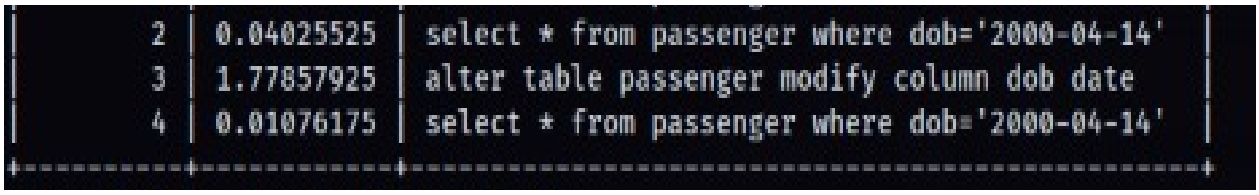
This image shows the number of row scans before and after changing the data type to tinyint.

Question-4:

Table: Passenger

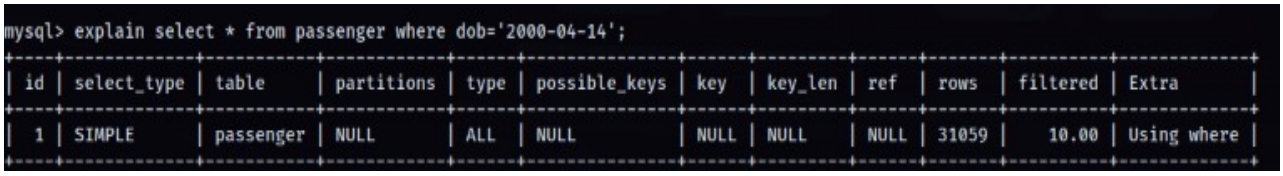
Attributes: Date_of_Birth(dob)

Justification: Initially we are fetching values when we have dob type as varchar.We are also fetching the values after the datatype then we are getting less time. When the data is organised in sequential file, the values initially stored based on the characters of the “dob” attribute. Later, they got stored in a different manner. They are stored initially based on sorted order of year, then month and then day. Hence, runtime is reduced.



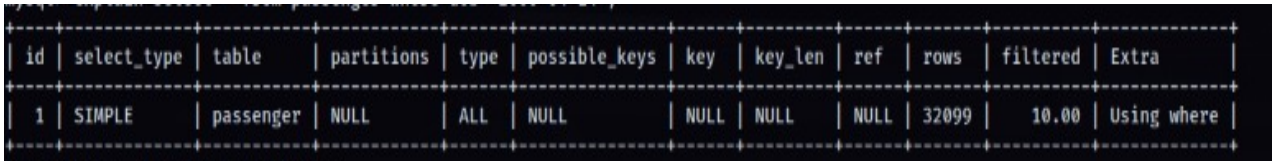
```
|        2 | 0.04025525 | select * from passenger where dob='2000-04-14' |
|        3 | 1.77857925 | alter table passenger modify column dob date   |
|        4 | 0.01076175 | select * from passenger where dob='2000-04-14' |
```

This image shows the decrease in the execution time of search after changing the data type of the column to date.



```
mysql> explain select * from passenger where dob='2000-04-14';
+----+-------------+-----------+------------+------+---------------+------+---------+------+-------+----------+-------------+
| id | select_type | table     | partitions | type | possible_keys | key  | key_len | ref  | rows  | filtered | Extra       |
+----+-------------+-----------+------------+------+---------------+------+---------+------+-------+----------+-------------+
|  1 | SIMPLE      | passenger | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 31059 |    10.00 | Using where |
+----+-------------+-----------+------------+------+---------------+------+---------+------+-------+----------+-------------+
```

This image shows the number of row scans for search before changing the data structure to date.



```
+----+-------------+-----------+------------+------+---------------+------+---------+------+-------+----------+-------------+
| id | select_type | table     | partitions | type | possible_keys | key  | key_len | ref  | rows  | filtered | Extra       |
+----+-------------+-----------+------------+------+---------------+------+---------+------+-------+----------+-------------+
|  1 | SIMPLE      | passenger | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 32099 |    10.00 | Using where |
+----+-------------+-----------+------------+------+---------------+------+---------+------+-------+----------+-------------+
```

This image shows the number of row scans for search after changing the data structure to date.

Question-5:

Table: Passenger

Attributes: Last Name

Justification: Initially, we took the count of the last name from the passenger without any null values. We immediately exported the data from the passenger table and made around ⅓ rd of the last names to NULL then we imported the newly made data into another copy table of passenger (passenger_q5). We then recalculated the count of last_name from passenger_q5. We got less computation time when there are NULL values and mysql is counting them as 0. As required, the count is total number of rows-number of rows which has NULL values.

Let us assume information is stored in B+ tree. The number of leaves OF B+ tree in case of the data with leave values is NULL is relatively less compared to no NULL values present in the tree. It simply shows us that the tree size is less when it has NULL values compared to the tree

having not having NULL values. Hence, searching time reduces.

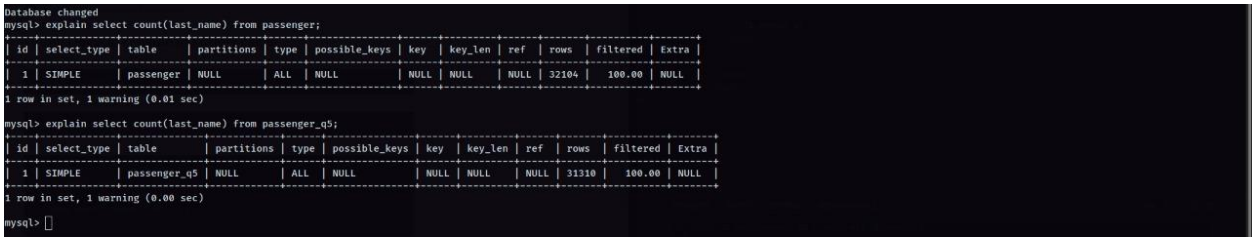
```
mysql> select count(*) from passenger;
+----------+
| count(*) |
+----------+
|    31783 |
+----------+
1 row in set (0.02 sec)

mysql> select count(last_name) from passenger_q5;
+------------------+
| count(last_name) |
+------------------+
|            21785 |
+------------------+
1 row in set (0.02 sec)

mysql> select count(last_name) from passenger;
+------------------+
| count(last_name) |
+------------------+
|            31783 |
+------------------+
1 row in set (0.01 sec)

mysql> show profiles;
+----------+------------+-----------------------------------------+
| Query_ID | Duration   | Query                                   |
+----------+------------+-----------------------------------------+
|        1 | 0.01048525 | select count(*) from passenger_q5       |
|        2 | 0.01593625 | select count(*) from passenger          |
|        3 | 0.02940375 | select count(last_name) from passenger_q5 |
|        4 | 0.01446475 | select count(last_name) from passenger  |
+----------+------------+-----------------------------------------+
4 rows in set, 1 warning (0.00 sec)

mysql>
```



```
Database changed
mysql> explain select count(last_name) from passenger;
+----+-------------+-----------+------------+------+---------------+------+---------+------+-------+----------+-------+
| id | select_type | table     | partitions | type | possible_keys | key  | key_len | ref  | rows  | filtered | Extra |
+----+-------------+-----------+------------+------+---------------+------+---------+------+-------+----------+-------+
|  1 | SIMPLE      | passenger | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 32104 |   100.00 | NULL  |
+----+-------------+-----------+------------+------+---------------+------+---------+------+-------+----------+-------+
1 row in set, 1 warning (0.01 sec)

mysql> explain select count(last_name) from passenger_q5;
+----+-------------+--------------+------------+------+---------------+------+---------+------+-------+----------+-------+
| id | select_type | table        | partitions | type | possible_keys | key  | key_len | ref  | rows  | filtered | Extra |
+----+-------------+--------------+------------+------+---------------+------+---------+------+-------+----------+-------+
|  1 | SIMPLE      | passenger_q5 | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 31310 |   100.00 | NULL  |
+----+-------------+--------------+------------+------+---------------+------+---------+------+-------+----------+-------+
1 row in set, 1 warning (0.00 sec)

mysql>
```

Question6:-

Tables: Transaction, train and Passenger

Attributes: *

Justification: Please note that MySql 8.2 version will not support query caching.

When we are trying to fetch the information using the same query more frequently, output of the query is stored in the cache. Output is stored in the cache based on two properties. How frequently we are calling the data and how huge the data is?. Basically, more the frequency, more the chance of storing and less the size of the output more the chance of storing.
This is our prediction, in case if MySql supports:
Total number of rows in Passenger, transaction and worker are 31783, 22687,3948 respectively. So the max_storage occupied is in the order of Passenger > transaction > worker.

Question-7:-

Table: Passenger and Transaction

Attributes: We use a natural join

Justification: Joins are better than executing multiple subqueries or multiple queries. This is because, while using multiple queries, there will be more number of comparisons and operations needed. But, a join will reduce these and give an optimized method to run these kind of queries. So, overally, joins decrease the burden of more comparisons and more data transfers of multiple queries by maing it a single query.

Multiple joins can become a problem for the server while designing the execution plan. Joining more number of trees givens more number of possible ways to join trees. This will pose a challenge while selecting the optimal way to join these tables. So, a good quality plan is selected but not the best one. If the engine sits on finding the best possible plan, it can take a very huge amount of time. Also, it is better to split the joins as there will a lot of confusion while understanding the query containing multiple joins.

```
mysql> show profiles;
+----------+------------+-----------------------------------------------------------------------------------------------------------+
| Query_ID | Duration   | Query                                                                                                     |
+----------+------------+-----------------------------------------------------------------------------------------------------------+
|        1 | 0.06802300 | select * from passenger where aadhar_no in(select aadhar_no from transact where amount>150)               |
|        2 | 0.06365050 | select aadhar_no,first_name,last_name,dob from passenger natural join transact where amount>150           |
+----------+------------+-----------------------------------------------------------------------------------------------------------+
```

This image shows the decrease in the time after using join.

```
mysql> explain select * from passenger where aadhar_no in(select aadhar_no from transact where amount>150);
+----+-------------+-------------+------------+--------+---------------+---------+---------+-----------------------+-------+----------+-------------+
| id | select_type | table       | partitions | type   | possible_keys | key     | key_len | ref                   | rows  | filtered | Extra       |
+----+-------------+-------------+------------+--------+---------------+---------+---------+-----------------------+-------+----------+-------------+
|  1 | SIMPLE      | <subquery2> | NULL       | ALL    | NULL          | NULL    | NULL    | NULL                  |  NULL |   100.00 | Using where |
|  1 | SIMPLE      | passenger   | NULL       | eq_ref | PRIMARY       | PRIMARY | 4       | <subquery2>.aadhar_no |     1 |   100.00 | NULL        |
|  2 | MATERIALIZED | transact   | NULL       | ALL    | aadhar_no     | NULL    | NULL    | NULL                  | 20665 |    33.33 | Using where |
+----+-------------+-------------+------------+--------+---------------+---------+---------+-----------------------+-------+----------+-------------+
3 rows in set, 1 warning (0.00 sec)

mysql> explain select aadhar_no,first_name,last_name,dob from passenger natural join transact where amount>150;
+----+-------------+-----------+------------+--------+---------------+---------+---------+-------------------------+-------+----------+-------------+
| id | select_type | table     | partitions | type   | possible_keys | key     | key_len | ref                     | rows  | filtered | Extra       |
+----+-------------+-----------+------------+--------+---------------+---------+---------+-------------------------+-------+----------+-------------+
|  1 | SIMPLE      | transact  | NULL       | ALL    | aadhar_no     | NULL    | NULL    | NULL                    | 20665 |    33.33 | Using where |
|  1 | SIMPLE      | passenger | NULL       | eq_ref | PRIMARY       | PRIMARY | 4       | railway.transact.aadhar_no |     1 |   100.00 | NULL        |
+----+-------------+-----------+------------+--------+---------------+---------+---------+-------------------------+-------+----------+-------------+
```

This image shows the number of row scans for the simple subquery and after join.

Question 8:

QUERY 1) info about passengers who are deboarding at ahmedabad and from which train

Nested subquery:

```
SELECT p.first_name, p.last_name, tic.train_id FROM passenger AS p, ticket AS tic
WHERE p.aadhar_no = tic.aadhar_no AND tic.train_id IN
        (SELECT train_id FROM train
        WHERE dest_pt = "Ahmedabad Jn");
```

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	train	NULL	ALL	PRIMARY	NULL	NULL	NULL	69719	10.00	Using where
	1	SIMPLE	tic	NULL	ref	PRIMARY,train_id	train_id	4	railway.train.train_id	1	100.00	Using index
	1	SIMPLE	p	NULL	eq_ref	PRIMARY	PRIMARY	4	railway.tic.aadhar_no	1	100.00	NULL

Using joins:

```
SELECT p.first_name, p.last_name, tic.train_id FROM passenger AS p, ticket AS tic, train
```

WHERE p.aadhar_no = tic.aadhar_no AND tic.train_id = train.train_id AND train.dest_pt = "Ahmedabad Jn";

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	train	NULL	ALL	PRIMARY	NULL	NULL	NULL	69719	10.00	Using where
	1	SIMPLE	tic	NULL	ref	PRIMARY,train_id	train_id	4	railway.train.train_id	1	100.00	Using index
	1	SIMPLE	p	NULL	eq_ref	PRIMARY	PRIMARY	4	railway.tic.aadhar_no	1	100.00	NULL

RESULTS:

First query is the nested subquery whereas the second query uses joins

	Query_ID	Duration	Query
▶	1	0.00010375	SHOW WARNINGS
	2	0.02743925	SELECT p.first_name, p.last_name, tic.train_id FROM passenger AS p, ticket AS tic WHER...
	3	0.02188275	SELECT p.first_name, p.last_name, tic.train_id FROM passenger AS p, ticket AS tic, train ...

```
mysql> show profiles;
+----------+------------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------+
| Query_ID | Duration   | Query                                                                                                                                                           |
+----------+------------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------+
|        1 | 0.03718025 | SELECT p.first_name, p.last_name, tic.train_id FROM passenger AS p, ticket AS tic  WHERE p.aadhar_no = tic.aadhar_no AND tic.train_id IN  (SELECT train_id FROM train   WHERE dest_pt = "Ahmedabad Jn") |
|        2 | 0.03583725 | SELECT p.first_name, p.last_name, tic.train_id FROM passenger AS p, ticket AS tic, train WHERE p.aadhar_no = tic.aadhar_no AND tic.train_id = train.train_id AND train.dest_pt = "Ahmedabad Jn" |
|        3 | 0.03644700 | SELECT p.first_name, p.last_name, tic.train_id FROM passenger AS p, ticket AS tic  WHERE p.aadhar_no = tic.aadhar_no AND tic.train_id IN  (SELECT train_id FROM train   WHERE dest_pt = "Ahmedabad Jn") |
|        4 | 0.03592475 | SELECT p.first_name, p.last_name, tic.train_id FROM passenger AS p, ticket AS tic, train WHERE p.aadhar_no = tic.aadhar_no AND tic.train_id = train.train_id AND train.dest_pt = "Ahmedabad Jn" |
+----------+------------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------+
4 rows in set, 1 warning (0.00 sec)
```

Thus, nested subquery takes more time to execute.

QUERY 2) number of passengers who are travelling from Mumbai central to Ahmedabad and have ticket status as RAC

Nested subquery:

select count(*) from passenger as p, ticket as tic
where p.aadhar_no = tic.aadhar_no and tic.ticket_status = "RAC" and tic.train_id in
(select train_id from train
where start_pt = "Mumbai Central" and dest_pt = "Ahmedabad Jn");

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	train	NULL	ALL	PRIMARY	NULL	NULL	NULL	69719	1.00	Using where
	1	SIMPLE	tic	NULL	ref	PRIMARY,train_id	train_id	4	railway.train.train_id	1	10.00	Using where
	1	SIMPLE	p	NULL	eq_ref	PRIMARY	PRIMARY	4	railway.tic.aadhar_no	1	100.00	Using index

Using joins:

select count(*) from passenger as p, ticket as tic, train
where p.aadhar_no = tic.aadhar_no and tic.ticket_status = "RAC" and tic.train_id = train.train_id
and train.start_pt = "Mumbai Central" and train.dest_pt = "Ahmedabad Jn";

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	train	NULL	ALL	PRIMARY	NULL	NULL	NULL	69719	1.00	Using where
	1	SIMPLE	tic	train	ref	PRIMARY,train_id	train_id	4	railway.train.train_id	1	10.00	Using where
	1	SIMPLE	p	NULL	eq_ref	PRIMARY	PRIMARY	4	railway.tic.aadhar_no	1	100.00	Using index

RESULTS:
First query is the nested subquery whereas the second query uses joins

	Query_ID	Duration	Query
▶	1	0.00008150	SHOW WARNINGS
	2	0.06506925	select count(*) from passenger as p, ticket as tic where p.aadhar_no = tic.aadhar_no and...
	3	0.04165950	select count(*) from passenger as p, ticket as tic, train where p.aadhar_no = tic.aadhar_n...

Here also, the nested subquery takes more time to execute.

Query 3) The details of the passengers who are travelling with tickets that cost more than 150 Rupees.

Nested subquery:
select * from passenger where aadhar_no in(select aadhar_no from transaction where amount>150);

Using joins:
select aadhar_no,first_name,last_name,dob from passenger natural join transaction transaction where amount>150;

```
mysql> show profiles;
+----------+------------+-----------------------------------------------------------------------------------------------------+
| Query_ID | Duration   | Query                                                                                               |
+----------+------------+-----------------------------------------------------------------------------------------------------+
|        1 | 0.06802300 | select * from passenger where aadhar_no in(select aadhar_no from transact where amount>150)         |
|        2 | 0.06365050 | select aadhar_no,first_name,last_name,dob from passenger natural join transact where amount>150     |
+----------+------------+-----------------------------------------------------------------------------------------------------+
```

This shows the decrease in the runtime for join compared to subquery.

```
mysql> explain select * from passenger where aadhar_no in(select aadhar_no from transact where amount>150);
+----+-------------+------------+------------+--------+---------------+---------+---------+-------------------------+-------+----------+-------------+
| id | select_type | table      | partitions | type   | possible_keys | key     | key_len | ref                     | rows  | filtered | Extra       |
+----+-------------+------------+------------+--------+---------------+---------+---------+-------------------------+-------+----------+-------------+
|  1 | SIMPLE      | <subquery2>| NULL       | ALL    | NULL          | NULL    | NULL    | NULL                    |  NULL |   100.00 | Using where |
|  1 | SIMPLE      | passenger  | NULL       | eq_ref | PRIMARY       | PRIMARY | 4       | <subquery2>.aadhar_no   |     1 |   100.00 | NULL        |
|  2 | MATERIALIZED| transact   | NULL       | ALL    | aadhar_no     | NULL    | NULL    | NULL                    | 20635 |    33.33 | Using where |
+----+-------------+------------+------------+--------+---------------+---------+---------+-------------------------+-------+----------+-------------+
3 rows in set, 1 warning (0.00 sec)

mysql> explain select aadhar_no,first_name,last_name,dob from passenger natural join transact where amount>150;
+----+-------------+----------+------------+--------+---------------+---------+---------+-------------------------+-------+----------+-------------+
| id | select_type | table    | partitions | type   | possible_keys | key     | key_len | ref                     | rows  | filtered | Extra       |
+----+-------------+----------+------------+--------+---------------+---------+---------+-------------------------+-------+----------+-------------+
|  1 | SIMPLE      | transact | NULL       | ALL    | aadhar_no     | NULL    | NULL    | NULL                    | 20635 |    33.33 | Using where |
|  1 | SIMPLE      | passenger| NULL       | eq_ref | PRIMARY       | PRIMARY | 4       | railway.transact.aadhar_no |     1 |   100.00 | NULL        |
+----+-------------+----------+------------+--------+---------------+---------+---------+-------------------------+-------+----------+-------------+
2 rows in set, 1 warning (0.00 sec)
```

This images shows the number of scans for two queries. One with subqueries and other with joins.