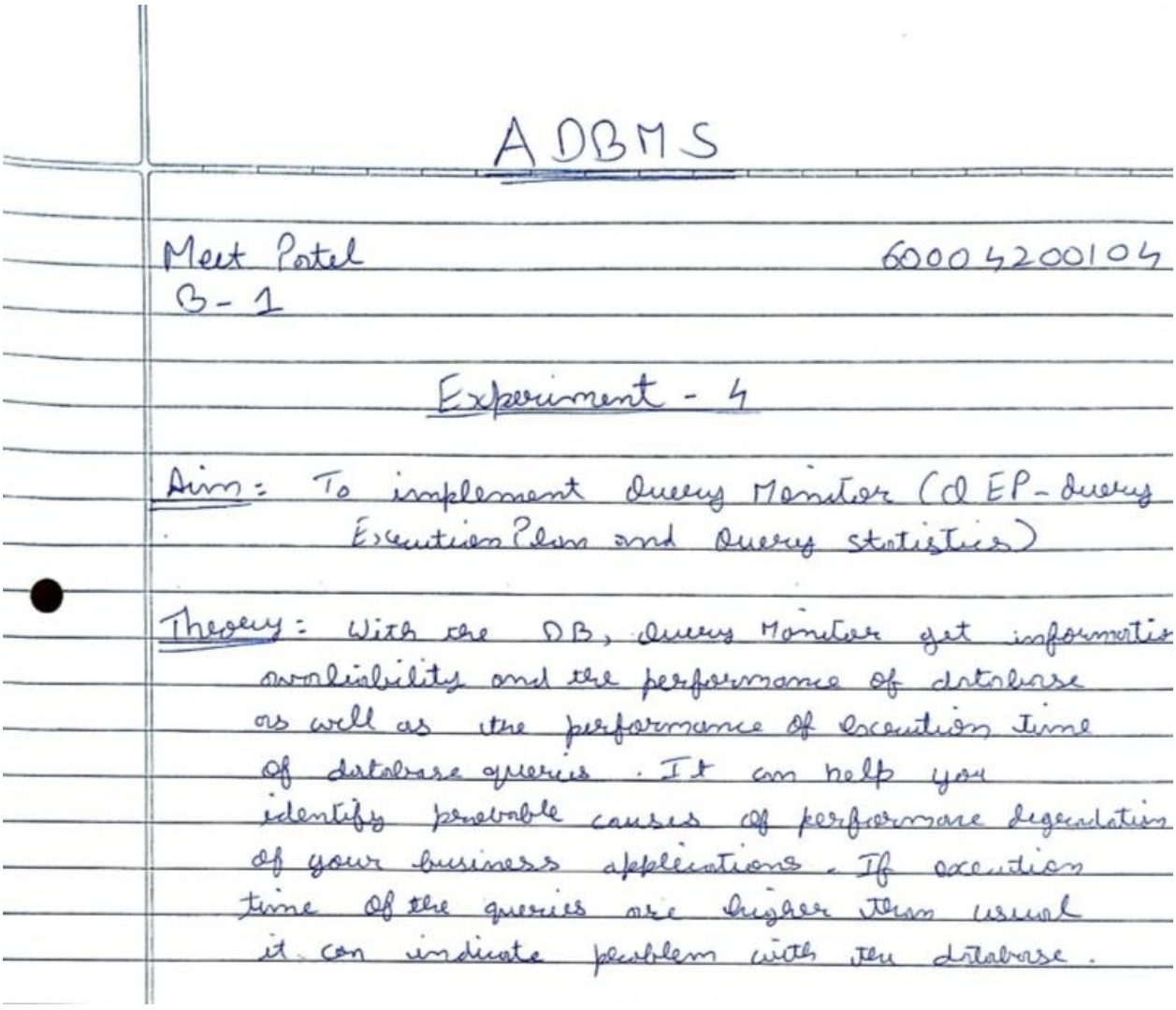
ADBMS

Name: Meet Patel SAP-ID: 60004200104

Division/Batch: B/B1 Branch: Computer Engineering

Experiment 4

AIM: To implement query monitor (query execution plan, query statistics)



ADBMS

Meet Patel 60004200104
B-1

Experiment - 4

Aim: To implement Query Monitor (QEP - Query Execution Plan and Query statistics)

Theory: With the DB, Query Monitor get information availability and the performance of database as well as the performance of execution time of database queries. It can help you identify probable causes of performance degradation of your business applications. If execution time of the queries are higher than usual it can indicate problem with the database.

Optimizing Queries with 'EXPLAIN'

Explain works with SELECT, DELETE, UPDATE, INSERT and REPLACE.

Explain is useful for examining queries involving partitioned tables.

When Explain is used with an explainable statement, MySQL displays information from optimizer about statement execution plan.

For select statement, explain produces additional execution plan information that can be displayed

Using show warnings.

With Explain you can also see where you should add indexes to tables so that statement executes faster by using indexes to find rows.

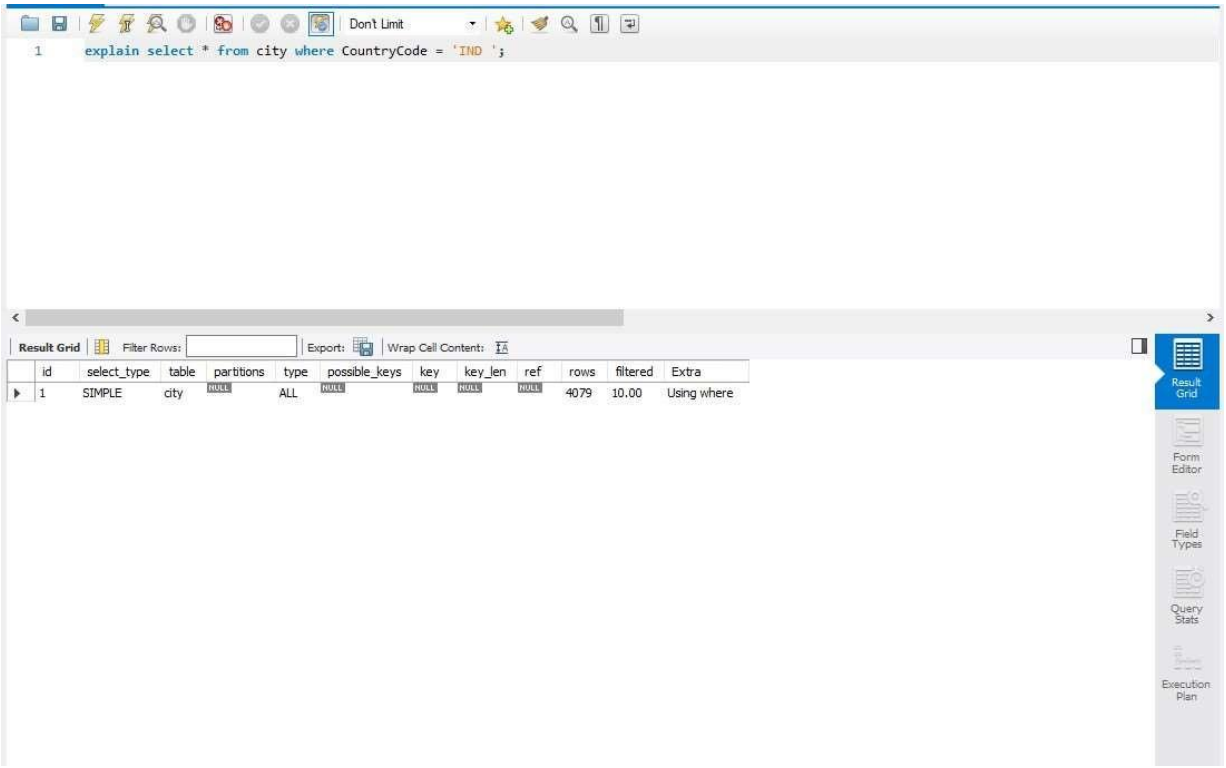
The optimizer trace may sometimes provide information complementary to that of Explain Explain can also be used to obtain information about the columns in a table.

Conclusion :-

The Query Monitor was implemented on various basis MySQL queries using explain keyword and the Query Execution Plan and Query statistics were observed and studied.

Implementation:

1) SELECT-FROM-WHERE



2) SELECT with GROUP BY-HAVING

3) AGGREGATE query

```
1 • explain select District, CountryCode, SUM(JSON_VALUE(Info, '$.Population')) as Population from city group by CountryCode order by Population;
2   #explain select *,JSON_VALUE(Info, '$.Population') as Population from city where CountryCode = 'IND' group by District having Population > 100000
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	city	NULL	ALL	NULL	NULL	NULL	NULL	4079	100.00	Using temporary; Using filesort

4) Nested query

```
1 • explain select Name,JSON_VALUE(Info, '$.Population') as Population from city where CountryCode = (Select CountryCode from countrylanguage where language = 'Hindi' and IsOfficial = 'T');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	city	NULL	ALL	NULL	NULL	NULL	NULL	4079	10.00	Using where
2	SUBQUERY	countrylanguage	NULL	ALL	NULL	NULL	NULL	NULL	984	5.00	Using where

5) JOIN

```
1 • explain select * from city natural join countrylanguage order by city.CountryCode;
2
3
4
5
6
7
8
9
10
11
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	city	NULL	ALL	NULL	NULL	NULL	NULL	4079	100.00	Using filesort
1	SIMPLE	countrylanguage	NULL	ref	PRIMARY,CountryCode	PRIMARY	12	world_x.city.CountryCode	1	100.00	NULL

Result Grid | Form Editor | Field Types | Query Stats

Conclusion:

Hence, Query monitoring was implemented successfully using MySQL.