Name: Meith Navlakha
SAP: 60004190068
ADBMS Practical

EXPERIMENT 1

Aim: Case study on professional and commercial database: Summary and Comparison.

Case Study on: E-Commerce website (own project)

Framework: PHP

Database: MySQL

MySQL is a relational database management system based on SQL – Structured Query Language. The application is used for a wide range of purposes, including data warehousing, e-commerce, and logging applications.

Used in association with a scripting language such as PHP or Perl (both offered on our hosting accounts) it is possible to create websites which will interact in real-time with a mySQL database to rapidly display categorised and searchable information to a website user.

MySQL provides many things, which are as follows:

- It provides the Referential Integrity between rows or columns of various tables.
- It allows us to implement database operations on tables, rows, columns, and indexes.
- It defines the database relationship in the form of tables (collection of rows and columns), also known as relations.

Features:

1) **Optimized Searching**

   I have used mySQL along with PHP for my project. For searching a product by category I have written a SQL query using WHERE clause that just searches into the entire product table that has all the other products as well. Thus, it has to traverse through the entire product table to fetch the specified category products tsking a long time for product information retrieval. making it highly time inefficient. Also, if there a minor spelling mistake then the search fails showing “no results found”.

Recommendation:

Currently I have used basic search for the website that just fetches the data when a query is fired. Instead I can use PHP with AJAX because it passes the request to an API that fetches the data and returns back in the JSON format. Firstly, this makes it platform independent as data is being received in JSON format. Also using AJAX we need not re-render the entire page. Only the segment of the page where data is being displayed is re-rendered. Thus, reducing the time and load on the server.

Secondly, can use ElasticSearch in Java for optimized searching. Elasticsearch uses a distributed system on top of Lucene StandardAnalyzer for indexing and automatic type guessing and utilizes a JSON based REST API to refer to Lucene features. Thus, using distributed system it much faster to fetch data than the basic SQL search which I implemented.

For the second problem the spelling mistake one, we can use node.js as it includes libraries that helps in coding **fuzzy search feature** i.e. if there’s a minor spelling mistake then it would direct to the most relevant route. This enhances the searching of our webpage. Also, node has tensorflow.js and other ML libraries which help in optimizing the searching featuring in the website.

2) Product Details Storing

MySQL uses RDBMS which in turn has rigidity with respect to the number of attributes for each instance(data record). I had made a table “Products” which had products of all the categories e.g. Necklace, Earing, Rings and so on where each category differs from the other in storing the dimensions. For example, Ring only requires one attribute “Diameter” while Earing requires attributes like “Size, Length and Height” and Category like Bridal Sets required even more. Thus, while storing the dimensions for different categories I had made different attributes into the tables which resulted into many empty cells for records of categories which did not have that attribute.

Recommendation:
To overcome this, we can use MongoDB being noSQL(non-relational) is more flexible with the schema and unstructured data. Thus, we can store data as key-value pair.
Moreover, mongoDB also supports storing value multiple datatypes we can just make a dimension key and store all the values in an array.

MySQL:
Product(<u>ProductID</u> , Name, ……. , Diameter, Size, Length, Height)

MongoDB:
Product
{
ProductID: Number,
Name: String,
….
Dimension : [Number],
}

3) Security

My project used MySQL which is highly vulnerable to SQL injection attacks. A hacker can skip the authentication process and manually inject SQL statements (or malicious payload) into the database. The database does not recognize the threat and executes the statement as if the application is sending the request.

Recommendation:

A) Certain measures that can be taken in order to lower the risk of an attack, such as using parameterized queries instead of a concatenated user input. Parameterized queries require the developers to define all the SQL code first and pass each parameter to the query later. This coding style enables the database to differentiate between code and data, a feature that is not possible with dynamic SQL queries. This is particularly important in WHERE clauses and INSERT or UPDATE statements.

B) Setting up a Firewall. Use a Web Application Firewall (WAF) to filter all traffic between web applications and your database. A firewall protects all web-facing apps, so blocks the SQLi and similar cyberattacks.

Conclusion:
In this experiment I have suggested recommendations on 3 features in my E-Commerce project based on PHP & mySQL. Firstly Optimized search where we can use AJAX and REST API for dynamic search instead of re-rendering the entire page, and using fuzzy search in node.js which increases the searching capacity for the user or use ElasticSearch in Java which improves the searching time by using distributed databases. Second feature was product details storing where SQL has schema rigidity while mongoDB is flexible with the data schema. Lastly security, since my project in SQL is vulnerable to SQL injection attacks we can implement parameterized queries or placeholders or use Web Application Firewall (WAF) to block the SQLi attacks.

Name: Meith Navlakha
SAP: 60004190068
ADBMS Practical

EXPERIEMNT 2

AIM: To explore different datasets and download a dataset for performing experiments in the further lab sessions.

DATASET:

Name: Wine Quality Dataset

Data Set Characteristics: Multivariate

Number of Instances: 4898

Number of Attributes: 12

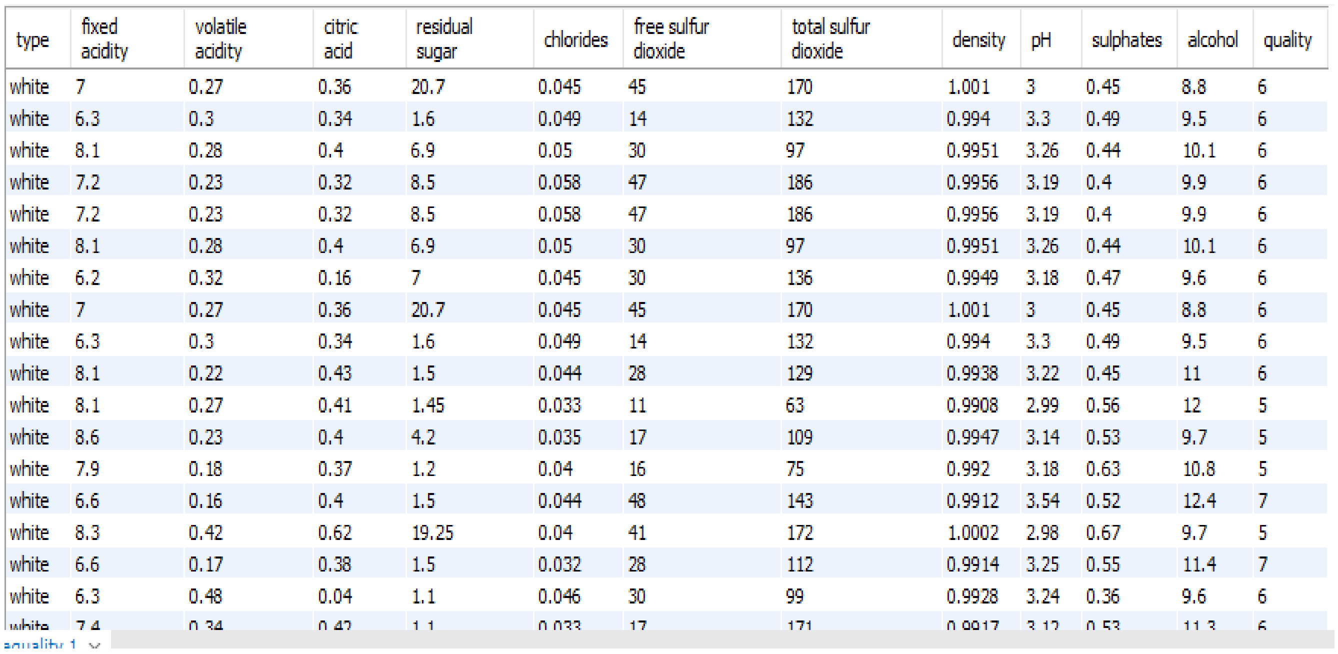
Attribute Information:

1 - fixed acidity
2 - volatile acidity
3 - citric acid
4 - residual sugar
5 - chlorides
6 - free sulfur dioxide
7 - total sulfur dioxide
8 - density
9 - pH
10 - sulphates
11 - alcohol
12 - quality (score between 0 and 10)

Name: Meith Navlakha
SAP: 60004190068
ADBMS Practical

OUTPUT:

Wine quality dataset



type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
white	7	0.27	0.36	20.7	0.045	45	170	1.001	3	0.45	8.8	6
white	6.3	0.3	0.34	1.6	0.049	14	132	0.994	3.3	0.49	9.5	6
white	8.1	0.28	0.4	6.9	0.05	30	97	0.9951	3.26	0.44	10.1	6
white	7.2	0.23	0.32	8.5	0.058	47	186	0.9956	3.19	0.4	9.9	6
white	7.2	0.23	0.32	8.5	0.058	47	186	0.9956	3.19	0.4	9.9	6
white	8.1	0.28	0.4	6.9	0.05	30	97	0.9951	3.26	0.44	10.1	6
white	6.2	0.32	0.16	7	0.045	30	136	0.9949	3.18	0.47	9.6	6
white	7	0.27	0.36	20.7	0.045	45	170	1.001	3	0.45	8.8	6
white	6.3	0.3	0.34	1.6	0.049	14	132	0.994	3.3	0.49	9.5	6
white	8.1	0.22	0.43	1.5	0.044	28	129	0.9938	3.22	0.45	11	6
white	8.1	0.27	0.41	1.45	0.033	11	63	0.9908	2.99	0.56	12	5
white	8.6	0.23	0.4	4.2	0.035	17	109	0.9947	3.14	0.53	9.7	5
white	7.9	0.18	0.37	1.2	0.04	16	75	0.992	3.18	0.63	10.8	5
white	6.6	0.16	0.4	1.5	0.044	48	143	0.9912	3.54	0.52	12.4	7
white	8.3	0.42	0.62	19.25	0.04	41	172	1.0002	2.98	0.67	9.7	5
white	6.6	0.17	0.38	1.5	0.032	28	112	0.9914	3.25	0.55	11.4	7
white	6.3	0.48	0.04	1.1	0.046	30	99	0.9928	3.24	0.36	9.6	6
white	7.4	0.34	0.42	1.1	0.033	17	171	0.9917	3.12	0.53	11.3	6

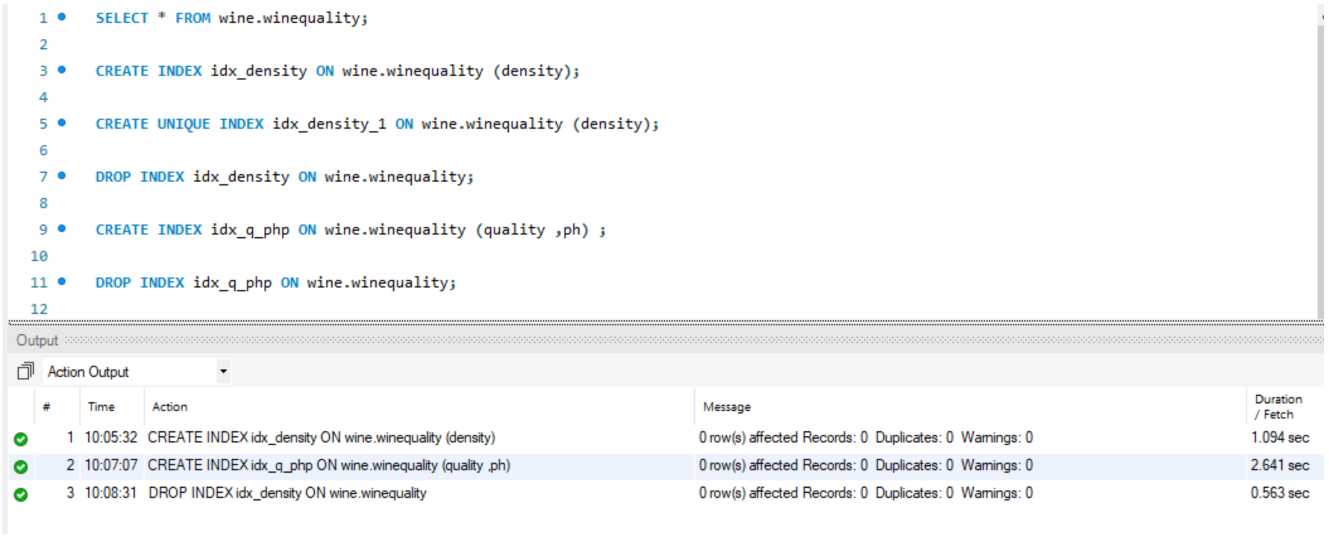
CONCLUSION: In this experiment, I downloaded Wine Quality Dataset from Kaggle website. The dataset was in the excel (i.e. .csv format) which I imported into the Workbench. During the import there were datatypes conflict and data incompatibility issues between the csv format and the workbench(sql) but those were resolved. The screenshot of the dataset is added above.

Name: Meith Navlakha
SAP: 60004190068
ADBMS Practical

EXPERIEMNT 3

AIM: To implement indexing on our selected dataset

PERFORMANCE:

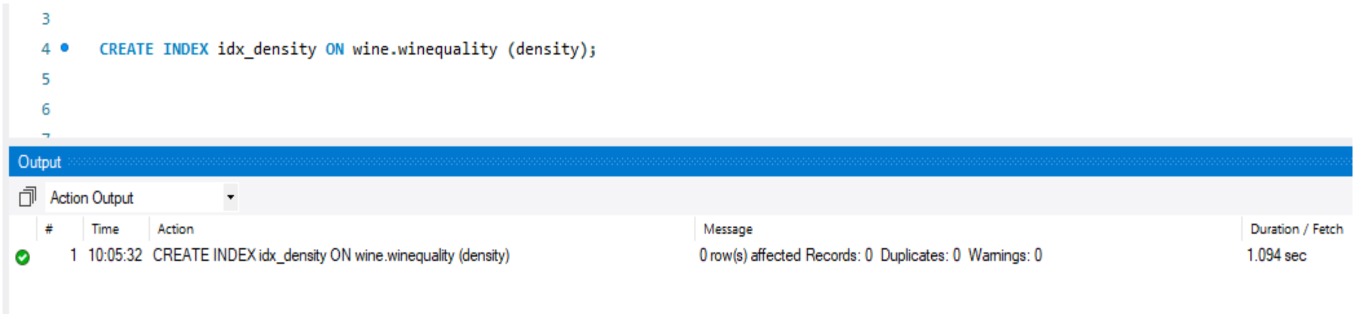


```
SELECT * FROM wine.winequality;

CREATE INDEX idx_density ON wine.winequality (density);

CREATE UNIQUE INDEX idx_density_1 ON wine.winequality (density);

DROP INDEX idx_density ON wine.winequality;

CREATE INDEX idx_q_php ON wine.winequality (quality ,ph) ;

DROP INDEX idx_q_php ON wine.winequality;

```

```
Output
Action Output
#  Time      Action                                                   Message                                                       Duration / Fetch
1  10:05:32  CREATE INDEX idx_density ON wine.winequality (density)   0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0      1.094 sec
2  10:07:07  CREATE INDEX idx_q_php ON wine.winequality (quality ,ph) 0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0      2.641 sec
3  10:08:31  DROP INDEX idx_density ON wine.winequality               0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0      0.563 sec
```

OUTPUT:

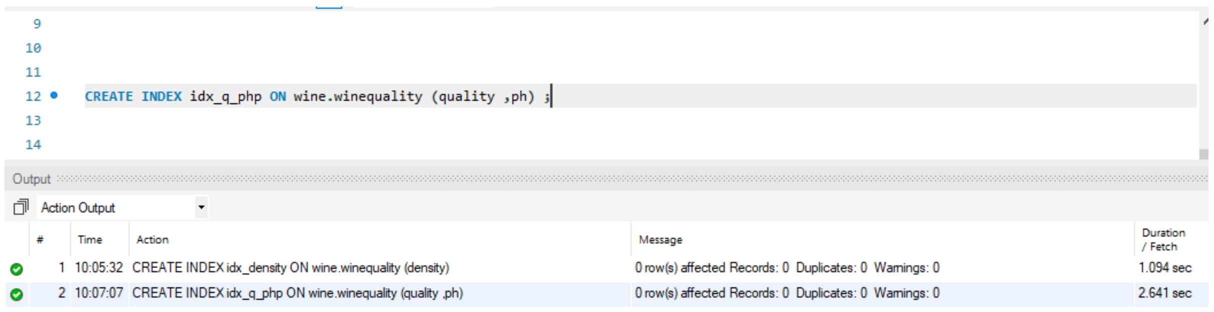
Created Index: idx_density



```
CREATE INDEX idx_density ON wine.winequality (density);
```

```
Output
Action Output
#  Time      Action                                                   Message                                                       Duration / Fetch
1  10:05:32  CREATE INDEX idx_density ON wine.winequality (density)   0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0      1.094 sec
```

Created Index: idx_q_php

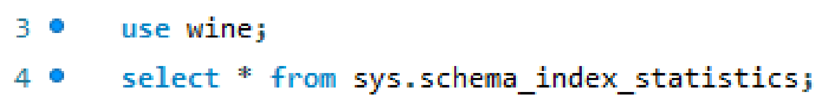


```
CREATE INDEX idx_q_php ON wine.winequality (quality ,ph) ;
```

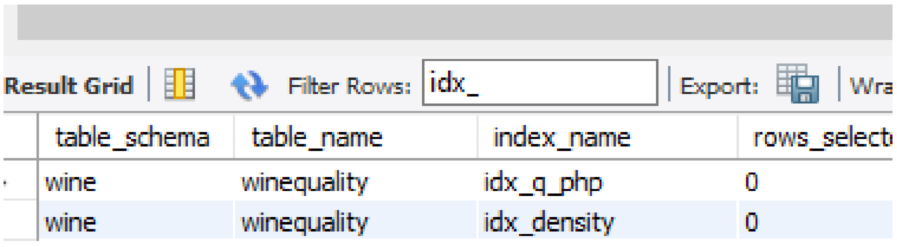
```
Output
Action Output
#  Time      Action                                                   Message                                                       Duration / Fetch
1  10:05:32  CREATE INDEX idx_density ON wine.winequality (density)   0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0      1.094 sec
2  10:07:07  CREATE INDEX idx_q_php ON wine.winequality (quality ,ph) 0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0      2.641 sec
```

Name: Meith Navlakha
SAP: 60004190068
ADBMS Practical

OUTPUT: Index Created

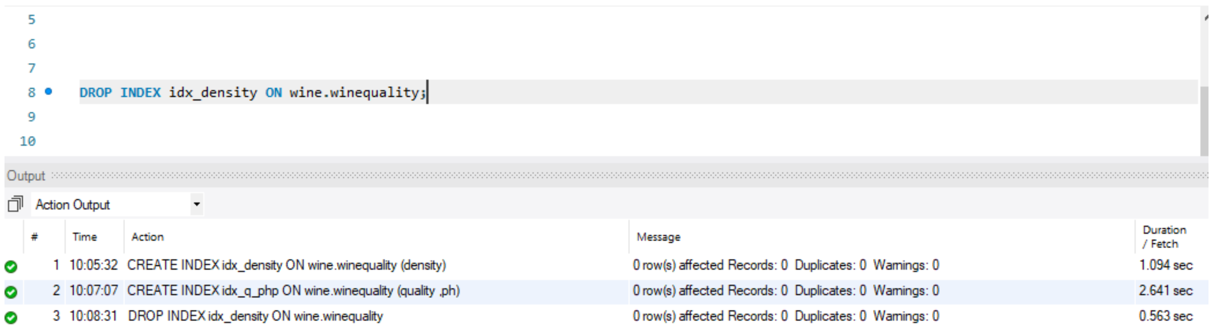


```
use wine;
select * from sys.schema_index_statistics;
```



table_schema	table_name	index_name	rows_select
wine	winequality	idx_q_php	0
wine	winequality	idx_density	0

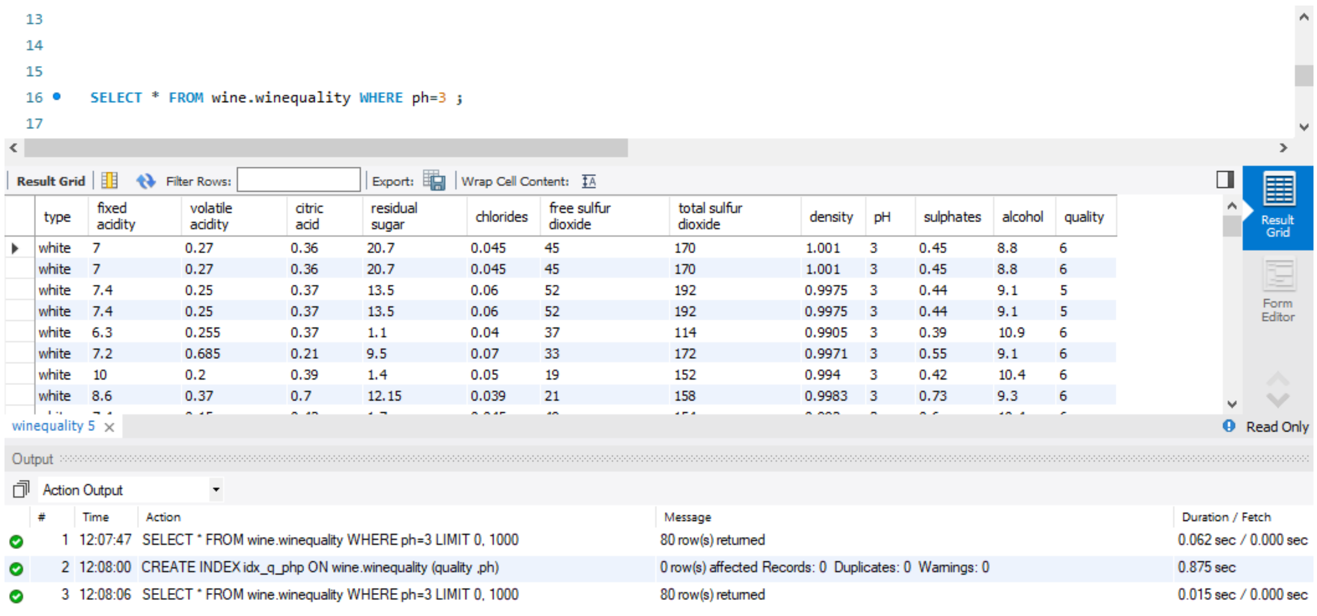
Drop idx_density



```
DROP INDEX idx_density ON wine.winequality;
```

```
#  Time      Action                                                         Message                                                     Duration / Fetch
1  10:05:32  CREATE INDEX idx_density ON wine.winequality (density)         0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0     1.094 sec
2  10:07:07  CREATE INDEX idx_q_php ON wine.winequality (quality ,ph)      0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0     2.641 sec
3  10:08:31  DROP INDEX idx_density ON wine.winequality                     0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0     0.563 sec
```

OBSERVATION:



```
SELECT * FROM wine.winequality WHERE ph=3 ;
```

type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
white	7	0.27	0.36	20.7	0.045	45	170	1.001	3	0.45	8.8	6
white	7	0.27	0.36	20.7	0.045	45	170	1.001	3	0.45	8.8	6
white	7.4	0.25	0.37	13.5	0.06	52	192	0.9975	3	0.44	9.1	5
white	7.4	0.25	0.37	13.5	0.06	52	192	0.9975	3	0.44	9.1	5
white	6.3	0.255	0.37	1.1	0.04	37	114	0.9905	3	0.39	10.9	6
white	7.2	0.685	0.21	9.5	0.07	33	172	0.9971	3	0.55	9.1	6
white	10	0.2	0.39	1.4	0.05	19	152	0.994	3	0.42	10.4	6
white	8.6	0.37	0.7	12.15	0.039	21	158	0.9983	3	0.73	9.3	6

```
#  Time      Action                                                    Message                                                     Duration / Fetch
1  12:07:47  SELECT * FROM wine.winequality WHERE ph=3 LIMIT 0, 1000   80 row(s) returned                                          0.062 sec / 0.000 sec
2  12:08:00  CREATE INDEX idx_q_php ON wine.winequality (quality ,ph)  0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0     0.875 sec
3  12:08:06  SELECT * FROM wine.winequality WHERE ph=3 LIMIT 0, 1000   80 row(s) returned                                          0.015 sec / 0.000 sec
```

In the console, we can observe that earlier it took a duration of 0.062sec to fetch the data of ph=3 but after adding index it takes 0.015sec. Thus, the overall duration of the process has decreased. The decrease is less in Wine dataset as the number of instances is quite small, there will be significant decrease in time for very large datasets.

CONCLUSION: In this experiment, I created 2 indexes ‘idx_density’ on density attribute and ‘idx_q_php’ on attributes quality followed by pH. For both the indexes, duplicate values are allowed and for restricting duplicate values we can add UNIQUE keyword in the query while creating the index. The overall data fetching speeds up, however, updating a table with indexes takes more time than updating a table without because the indexes also need an update. The implementation of the experiment was error free.

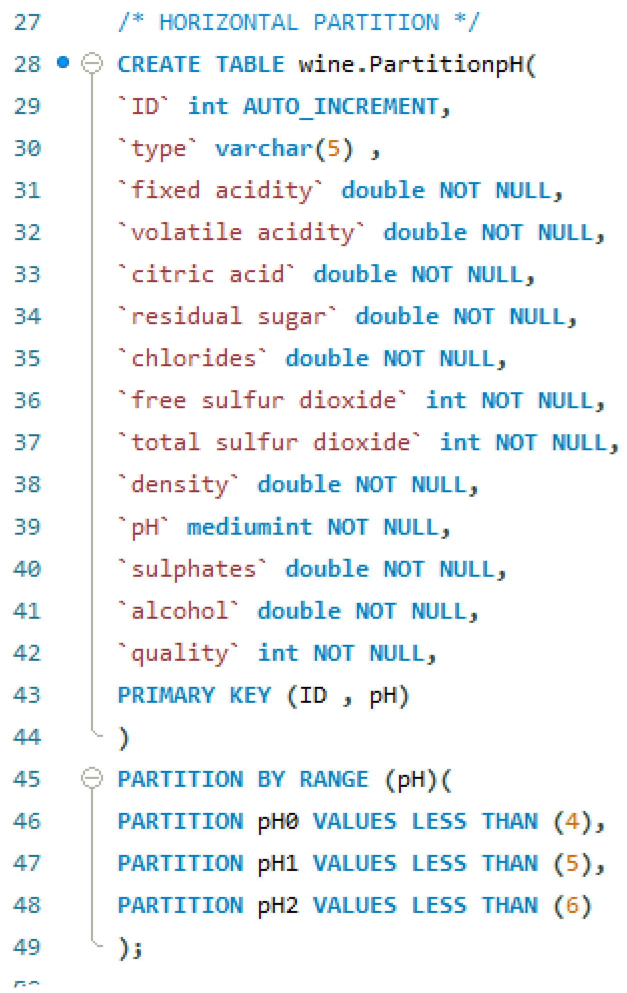
Name: Meith Navlakha
SAP: 60004190068
ADBMS Practical

EXPERIEMNT 4

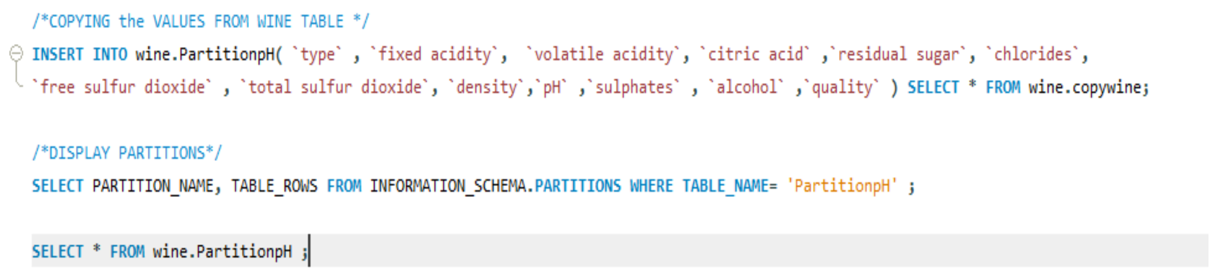
AIM: To implement Fragmentation on our selected dataset.

PERFORMANCE:

1) Horizontal Partitioning:



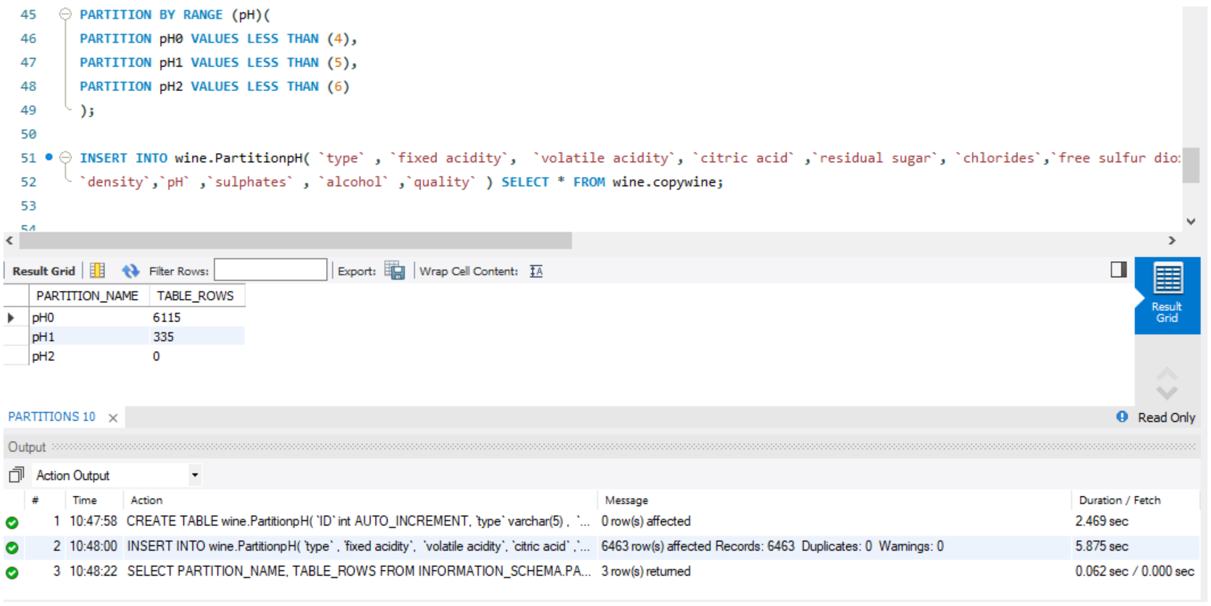
```
/* HORIZONTAL PARTITION */
CREATE TABLE wine.PartitionpH(
`ID` int AUTO_INCREMENT,
`type` varchar(5) ,
`fixed acidity` double NOT NULL,
`volatile acidity` double NOT NULL,
`citric acid` double NOT NULL,
`residual sugar` double NOT NULL,
`chlorides` double NOT NULL,
`free sulfur dioxide` int NOT NULL,
`total sulfur dioxide` int NOT NULL,
`density` double NOT NULL,
`pH` mediumint NOT NULL,
`sulphates` double NOT NULL,
`alcohol` double NOT NULL,
`quality` int NOT NULL,
PRIMARY KEY (ID , pH)
)
PARTITION BY RANGE (pH)(
PARTITION pH0 VALUES LESS THAN (4),
PARTITION pH1 VALUES LESS THAN (5),
PARTITION pH2 VALUES LESS THAN (6)
);
```



```
/*COPYING the VALUES FROM WINE TABLE */
INSERT INTO wine.PartitionpH( `type` , `fixed acidity`,  `volatile acidity`, `citric acid` ,`residual sugar`, `chlorides`,
`free sulfur dioxide` , `total sulfur dioxide`, `density`,`pH` ,`sulphates` , `alcohol` ,`quality` ) SELECT * FROM wine.copywine;

/*DISPLAY PARTITIONS*/
SELECT PARTITION_NAME, TABLE_ROWS FROM INFORMATION_SCHEMA.PARTITIONS WHERE TABLE_NAME= 'PartitionpH' ;

SELECT * FROM wine.PartitionpH ;
```

Name: Meith Navlakha
SAP: 60004190068
ADBMS Practical

OUTPUT:



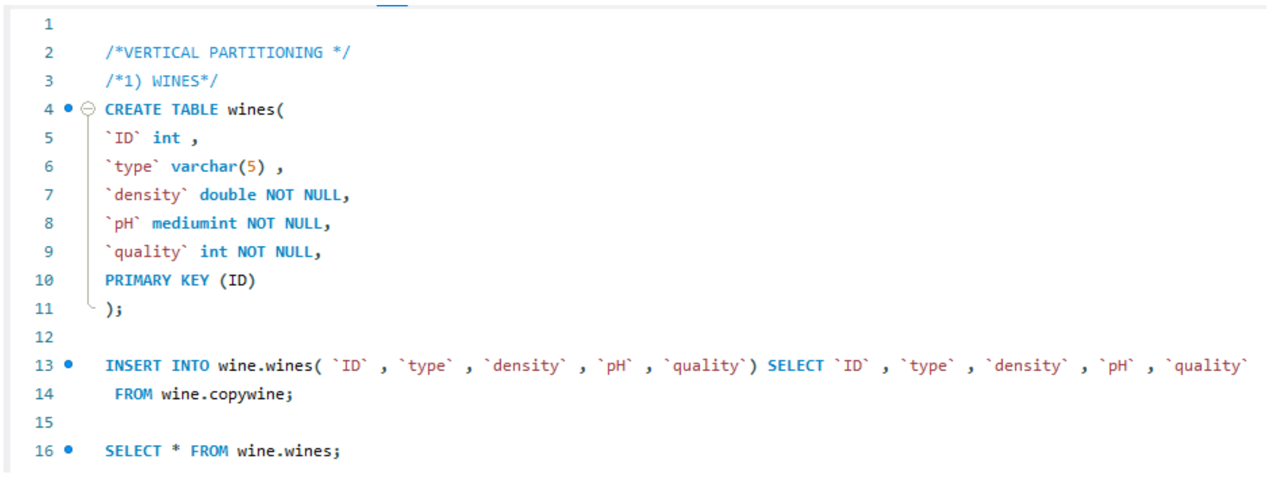
```
PARTITION BY RANGE (pH)(
PARTITION pH0 VALUES LESS THAN (4),
PARTITION pH1 VALUES LESS THAN (5),
PARTITION pH2 VALUES LESS THAN (6)
);

INSERT INTO wine.PartitionpH( `type` , `fixed acidity`,  `volatile acidity`, `citric acid` ,`residual sugar`, `chlorides`,`free sulfur dio:
`density`,`pH` ,`sulphates` , `alcohol` ,`quality` ) SELECT * FROM wine.copywine;
```

PARTITION_NAME	TABLE_ROWS
pH0	6115
pH1	335
pH2	0

```
PARTITIONS 10                                                                Read Only
Output
Action Output
 #  Time      Action                                                                     Message                                                       Duration / Fetch
 1  10:47:58  CREATE TABLE wine.PartitionpH( `ID` int AUTO_INCREMENT, `type` varchar(5) , `...  0 row(s) affected                                             2.469 sec
 2  10:48:00  INSERT INTO wine.PartitionpH( `type` , `fixed acidity`, `volatile acidity`, `citric acid` ,`...  6463 row(s) affected Records: 6463  Duplicates: 0  Warnings: 0  5.875 sec
 3  10:48:22  SELECT PARTITION_NAME, TABLE_ROWS FROM INFORMATION_SCHEMA.PA...              3 row(s) returned                                             0.062 sec / 0.000 sec
```

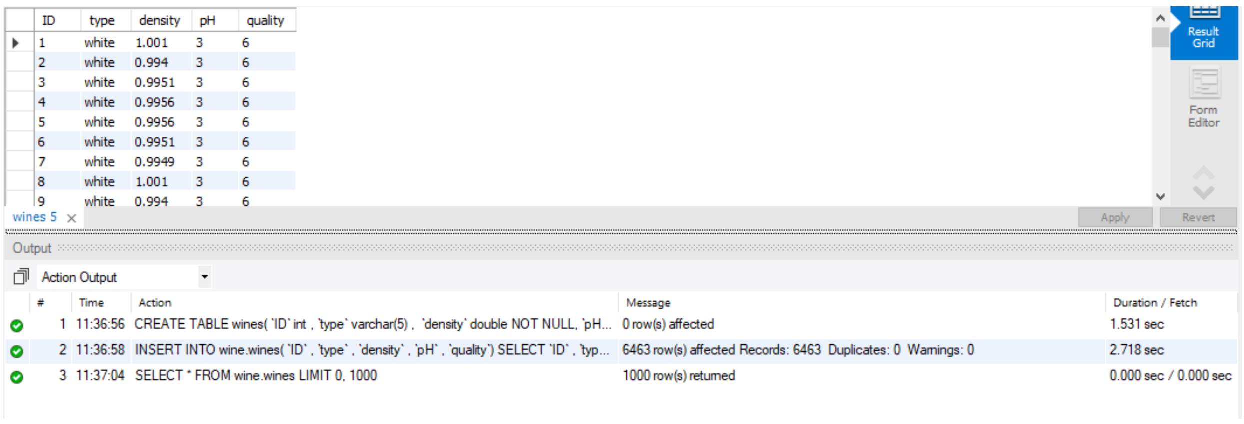
2) Vertical Partitioning:

A) Wines Table:



```

/*VERTICAL PARTITIONING */
/*1) WINES*/
CREATE TABLE wines(
`ID` int ,
`type` varchar(5) ,
`density` double NOT NULL,
`pH` mediumint NOT NULL,
`quality` int NOT NULL,
PRIMARY KEY (ID)
);

INSERT INTO wine.wines( `ID` , `type` , `density` , `pH` , `quality`) SELECT `ID` , `type` , `density` , `pH` , `quality`
 FROM wine.copywine;

SELECT * FROM wine.wines;
```

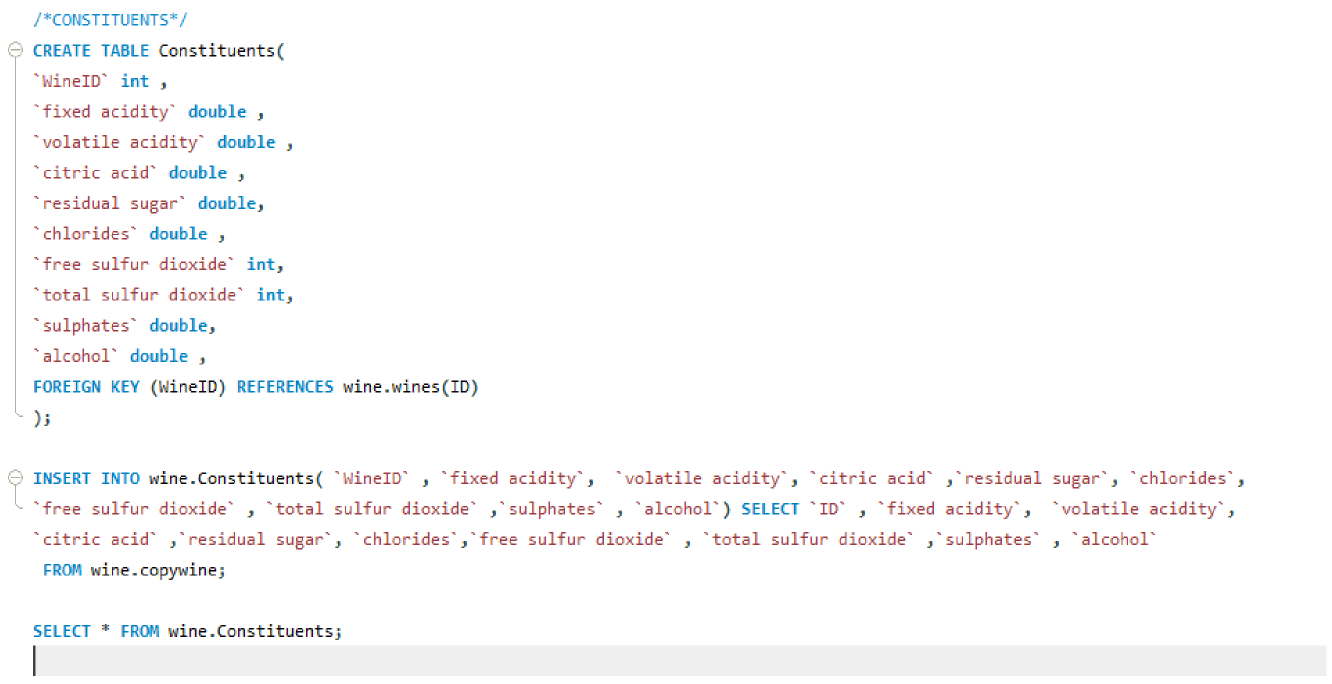
OUTPUT:



ID	type	density	pH	quality
1	white	1.001	3	6
2	white	0.994	3	6
3	white	0.9951	3	6
4	white	0.9956	3	6
5	white	0.9956	3	6
6	white	0.9951	3	6
7	white	0.9949	3	6
8	white	1.001	3	6
9	white	0.994	3	6

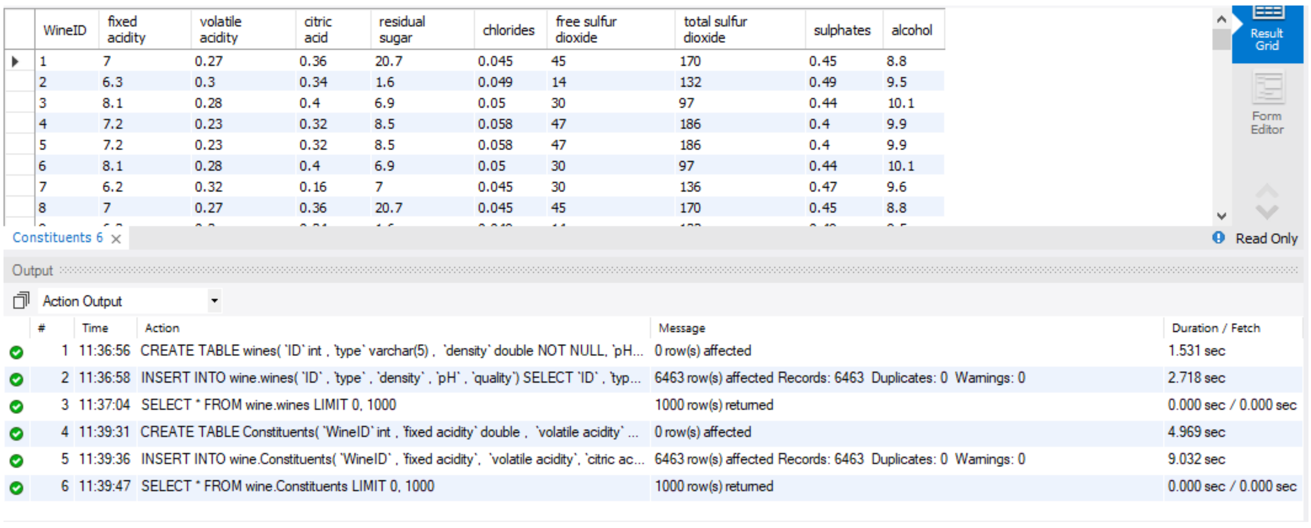
```
wines 5
Output
Action Output
 #  Time      Action                                                                     Message                                                       Duration / Fetch
 1  11:36:56  CREATE TABLE wines( `ID` int , `type` varchar(5) , `density` double NOT NULL, `pH...  0 row(s) affected                                   1.531 sec
 2  11:36:58  INSERT INTO wine.wines( `ID` , `type` , `density` , `pH` , `quality`) SELECT `ID` , `typ...  6463 row(s) affected Records: 6463  Duplicates: 0  Warnings: 0  2.718 sec
 3  11:37:04  SELECT * FROM wine.wines LIMIT 0, 1000                                     1000 row(s) returned                                          0.000 sec / 0.000 sec
```

B) Constituents Table



```
/*CONSTITUENTS*/
CREATE TABLE Constituents(
`WineID` int ,
`fixed acidity` double ,
`volatile acidity` double ,
`citric acid` double ,
`residual sugar` double,
`chlorides` double ,
`free sulfur dioxide` int,
`total sulfur dioxide` int,
`sulphates` double,
`alcohol` double ,
FOREIGN KEY (WineID) REFERENCES wine.wines(ID)
);

INSERT INTO wine.Constituents( `WineID` , `fixed acidity`,  `volatile acidity`, `citric acid` ,`residual sugar`, `chlorides`,
`free sulfur dioxide` , `total sulfur dioxide` ,`sulphates` , `alcohol`) SELECT `ID` , `fixed acidity`,  `volatile acidity`,
`citric acid` ,`residual sugar`, `chlorides`,`free sulfur dioxide` , `total sulfur dioxide` ,`sulphates` , `alcohol`
 FROM wine.copywine;

SELECT * FROM wine.Constituents;
```

OUTPUT:



	WineID	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	sulphates	alcohol
▶	1	7	0.27	0.36	20.7	0.045	45	170	0.45	8.8
	2	6.3	0.3	0.34	1.6	0.049	14	132	0.49	9.5
	3	8.1	0.28	0.4	6.9	0.05	30	97	0.44	10.1
	4	7.2	0.23	0.32	8.5	0.058	47	186	0.4	9.9
	5	7.2	0.23	0.32	8.5	0.058	47	186	0.4	9.9
	6	8.1	0.28	0.4	6.9	0.05	30	97	0.44	10.1
	7	6.2	0.32	0.16	7	0.045	30	136	0.47	9.6
	8	7	0.27	0.36	20.7	0.045	45	170	0.45	8.8

	#	Time	Action	Message	Duration / Fetch
✓	1	11:36:56	CREATE TABLE wines(`ID` int , `type` varchar(5) , `density` double NOT NULL, `pH...	0 row(s) affected	1.531 sec
✓	2	11:36:58	INSERT INTO wine.wines(`ID` , `type` , `density` , `pH` , `quality`) SELECT `ID` , `typ...	6463 row(s) affected Records: 6463 Duplicates: 0 Warnings: 0	2.718 sec
✓	3	11:37:04	SELECT * FROM wine.wines LIMIT 0, 1000	1000 row(s) returned	0.000 sec / 0.000 sec
✓	4	11:39:31	CREATE TABLE Constituents(`WineID` int , `fixed acidity` double , `volatile acidity` ...	0 row(s) affected	4.969 sec
✓	5	11:39:36	INSERT INTO wine.Constituents(`WineID` , `fixed acidity`, `volatile acidity`, `citric ac...	6463 row(s) affected Records: 6463 Duplicates: 0 Warnings: 0	9.032 sec
✓	6	11:39:47	SELECT * FROM wine.Constituents LIMIT 0, 1000	1000 row(s) returned	0.000 sec / 0.000 sec

CONCLUSION: In this experiment, I created firstly, Horizontal Partition on the wine quality table by grouping the data items with respect to the pH values. Partition pH0(less than 4), pH1(4-5) and pH2(5-6) are created and the entire partition is stored on a new table ‘partitionpH’ so as to preserve the original wine table. Secondly, I created Vertical Partitioning of the dataset into tables ‘wines’ and ‘Constituents’, where the ID of wines is linked with WineID of Constituents’.

Name: Meith Navlakha
SAP: 60004190068
ADBMS Practical

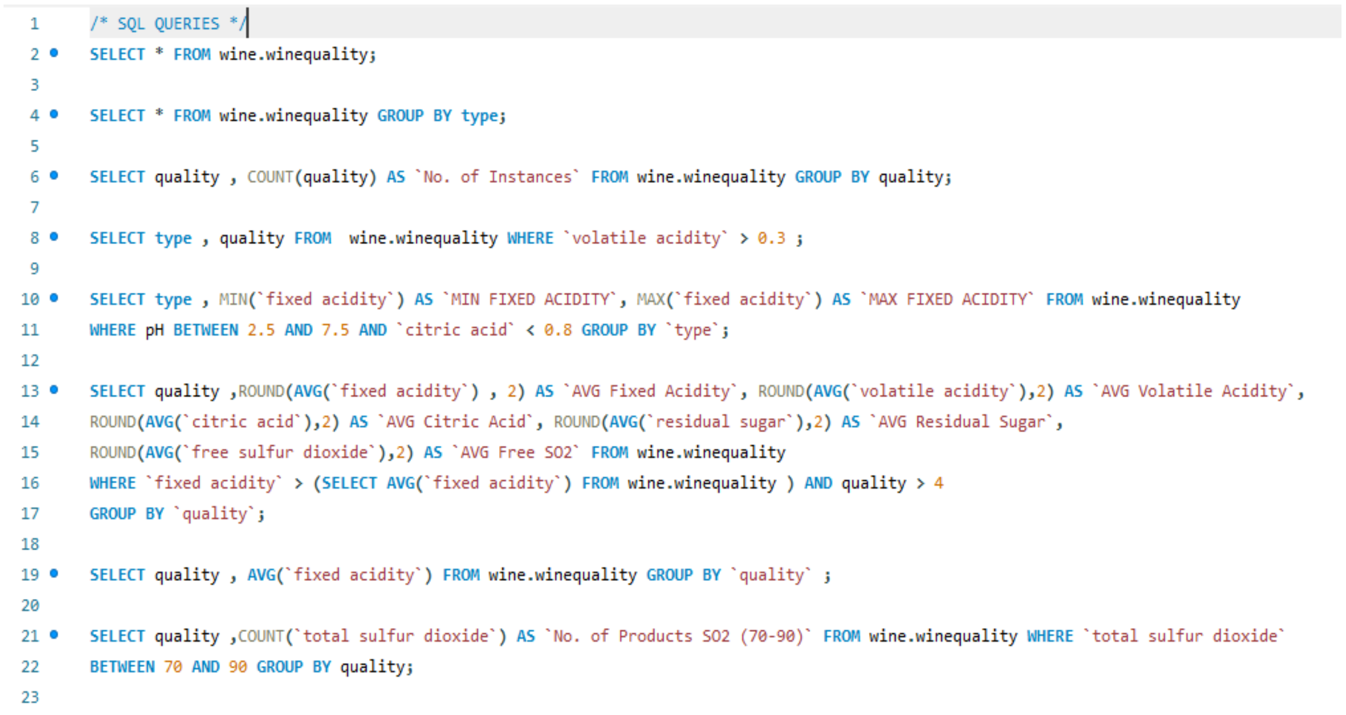
EXPERIEMNT 5

AIM: Perform basic SQL Queries on the selected dataset.

PERFORMANCE:

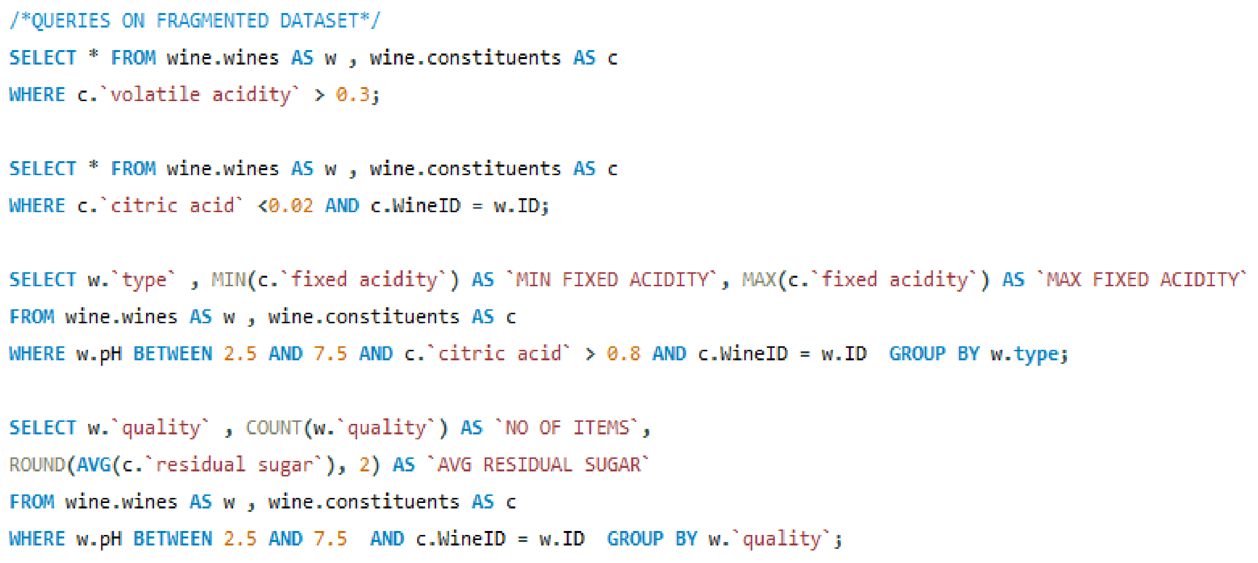
SQL QUERIES:

a) Original Dataset:



```sql
/* SQL QUERIES */
SELECT * FROM wine.winequality;

SELECT * FROM wine.winequality GROUP BY type;

SELECT quality , COUNT(quality) AS `No. of Instances` FROM wine.winequality GROUP BY quality;

SELECT type , quality FROM  wine.winequality WHERE `volatile acidity` > 0.3 ;

SELECT type , MIN(`fixed acidity`) AS `MIN FIXED ACIDITY`, MAX(`fixed acidity`) AS `MAX FIXED ACIDITY` FROM wine.winequality
WHERE pH BETWEEN 2.5 AND 7.5 AND `citric acid` < 0.8 GROUP BY `type`;

SELECT quality ,ROUND(AVG(`fixed acidity`) , 2) AS `AVG Fixed Acidity`, ROUND(AVG(`volatile acidity`),2) AS `AVG Volatile Acidity`,
ROUND(AVG(`citric acid`),2) AS `AVG Citric Acid`, ROUND(AVG(`residual sugar`),2) AS `AVG Residual Sugar`,
ROUND(AVG(`free sulfur dioxide`),2) AS `AVG Free SO2` FROM wine.winequality
WHERE `fixed acidity` > (SELECT AVG(`fixed acidity`) FROM wine.winequality ) AND quality > 4
GROUP BY `quality`;

SELECT quality , AVG(`fixed acidity`) FROM wine.winequality GROUP BY `quality` ;

SELECT quality ,COUNT(`total sulfur dioxide`) AS `No. of Products SO2 (70-90)` FROM wine.winequality WHERE `total sulfur dioxide`
BETWEEN 70 AND 90 GROUP BY quality;

```

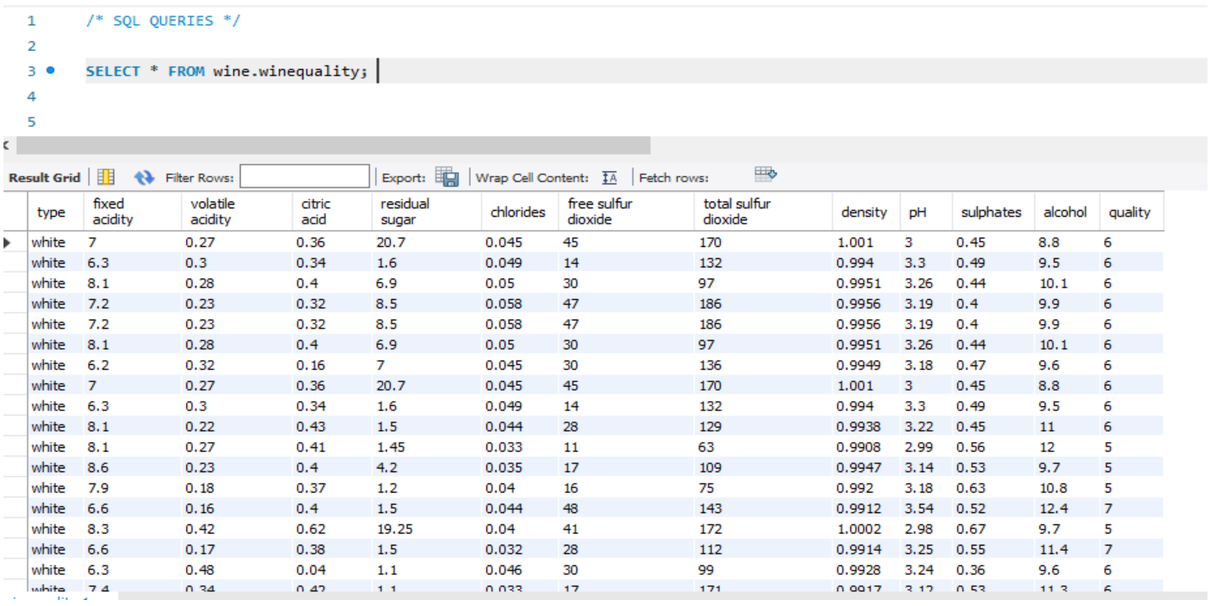
b) Fragmented Dataset:



```sql
/*QUERIES ON FRAGMENTED DATASET*/
SELECT * FROM wine.wines AS w , wine.constituents AS c
WHERE c.`volatile acidity` > 0.3;

SELECT * FROM wine.wines AS w , wine.constituents AS c
WHERE c.`citric acid` <0.02 AND c.WineID = w.ID;

SELECT w.`type`  , MIN(c.`fixed acidity`) AS `MIN FIXED ACIDITY`, MAX(c.`fixed acidity`) AS `MAX FIXED ACIDITY`
FROM wine.wines AS w , wine.constituents AS c
WHERE w.pH BETWEEN 2.5 AND 7.5 AND c.`citric acid` > 0.8 AND c.WineID = w.ID  GROUP BY w.type;

SELECT w.`quality` , COUNT(w.`quality`) AS `NO OF ITEMS`,
ROUND(AVG(c.`residual sugar`), 2) AS `AVG RESIDUAL SUGAR`
FROM wine.wines AS w , wine.constituents AS c
WHERE w.pH BETWEEN 2.5 AND 7.5  AND c.WineID = w.ID  GROUP BY w.`quality`;
```

Name: Meith Navlakha
SAP: 60004190068
ADBMS Practical

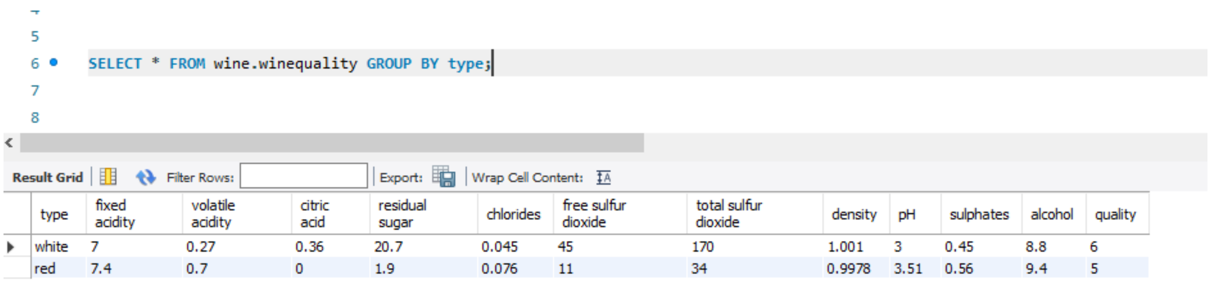
OUTPUT:



```sql
/* SQL QUERIES */

SELECT * FROM wine.winequality;
```

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

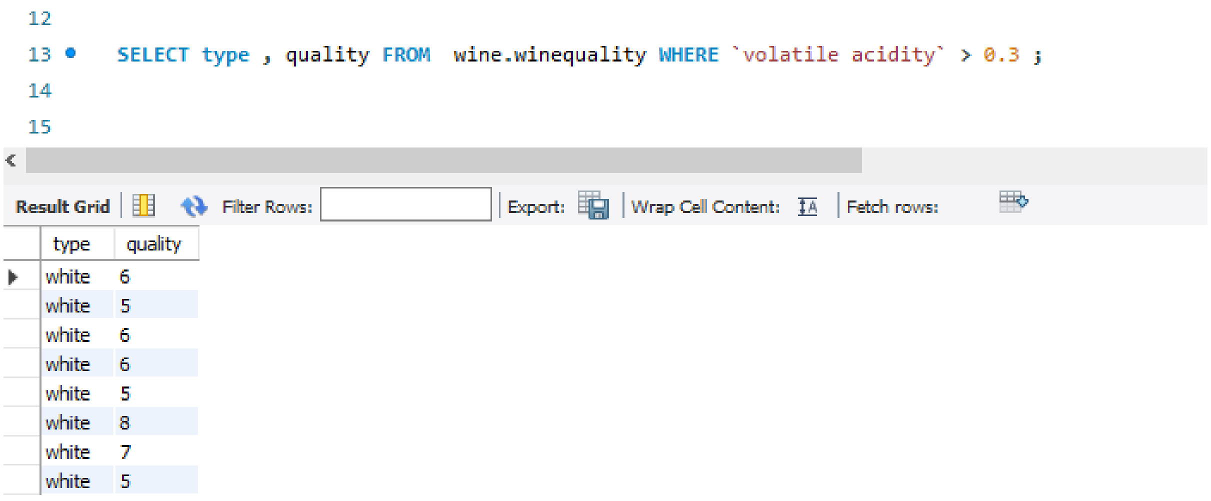
type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
white	7	0.27	0.36	20.7	0.045	45	170	1.001	3	0.45	8.8	6
white	6.3	0.3	0.34	1.6	0.049	14	132	0.994	3.3	0.49	9.5	6
white	8.1	0.28	0.4	6.9	0.05	30	97	0.9951	3.26	0.44	10.1	6
white	7.2	0.23	0.32	8.5	0.058	47	186	0.9956	3.19	0.4	9.9	6
white	7.2	0.23	0.32	8.5	0.058	47	186	0.9956	3.19	0.4	9.9	6
white	8.1	0.28	0.4	6.9	0.05	30	97	0.9951	3.26	0.44	10.1	6
white	6.2	0.32	0.16	7	0.045	30	136	0.9949	3.18	0.47	9.6	6
white	7	0.27	0.36	20.7	0.045	45	170	1.001	3	0.45	8.8	6
white	6.3	0.3	0.34	1.6	0.049	14	132	0.994	3.3	0.49	9.5	6
white	8.1	0.22	0.43	1.5	0.044	28	129	0.9938	3.22	0.45	11	6
white	8.1	0.27	0.41	1.45	0.033	11	63	0.9908	2.99	0.56	12	5
white	8.6	0.23	0.4	4.2	0.035	17	109	0.9947	3.14	0.53	9.7	5
white	7.9	0.18	0.37	1.2	0.04	16	75	0.992	3.18	0.63	10.8	5
white	6.6	0.16	0.4	1.5	0.044	48	143	0.9912	3.54	0.52	12.4	7
white	8.3	0.42	0.62	19.25	0.04	41	172	1.0002	2.98	0.67	9.7	5
white	6.6	0.17	0.38	1.5	0.032	28	112	0.9914	3.25	0.55	11.4	7
white	6.3	0.48	0.04	1.1	0.046	30	99	0.9928	3.24	0.36	9.6	6
white	7.4	0.34	0.42	1.1	0.033	17	171	0.9917	3.12	0.53	11.3	6



```sql
SELECT * FROM wine.winequality GROUP BY type;
```

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

type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
white	7	0.27	0.36	20.7	0.045	45	170	1.001	3	0.45	8.8	6
red	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5

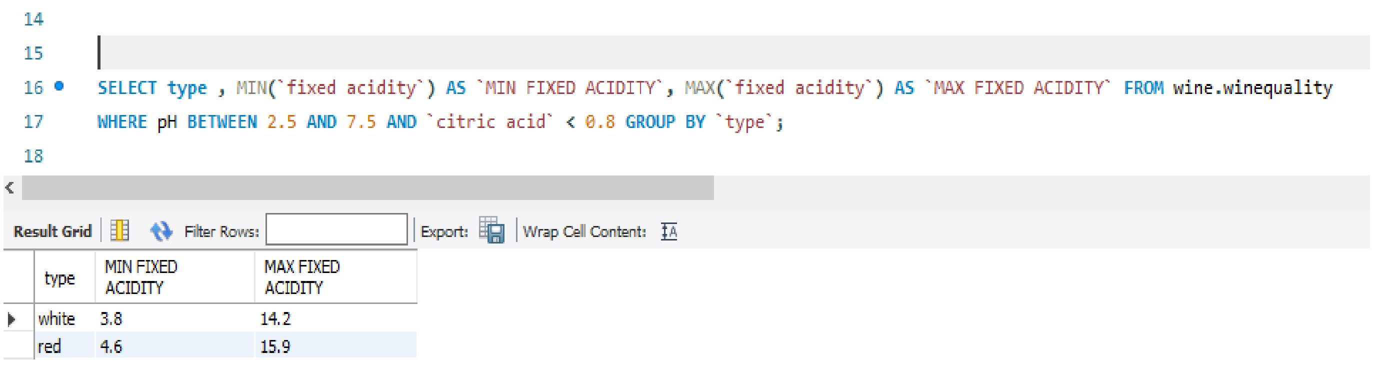


```sql
SELECT type , quality FROM  wine.winequality WHERE `volatile acidity` > 0.3 ;
```

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

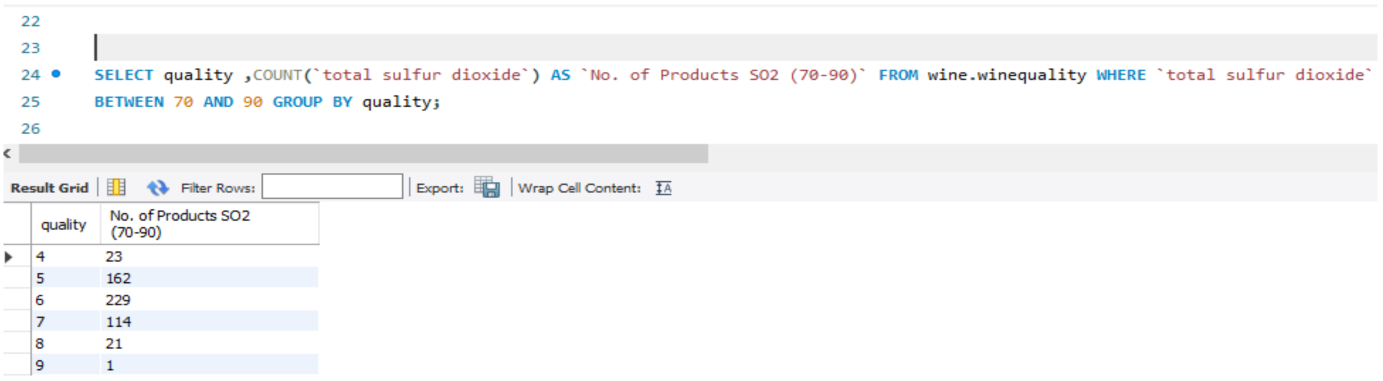
type	quality
white	6
white	5
white	6
white	6
white	5
white	8
white	7
white	5

Name: Meith Navlakha
SAP: 60004190068
ADBMS Practical



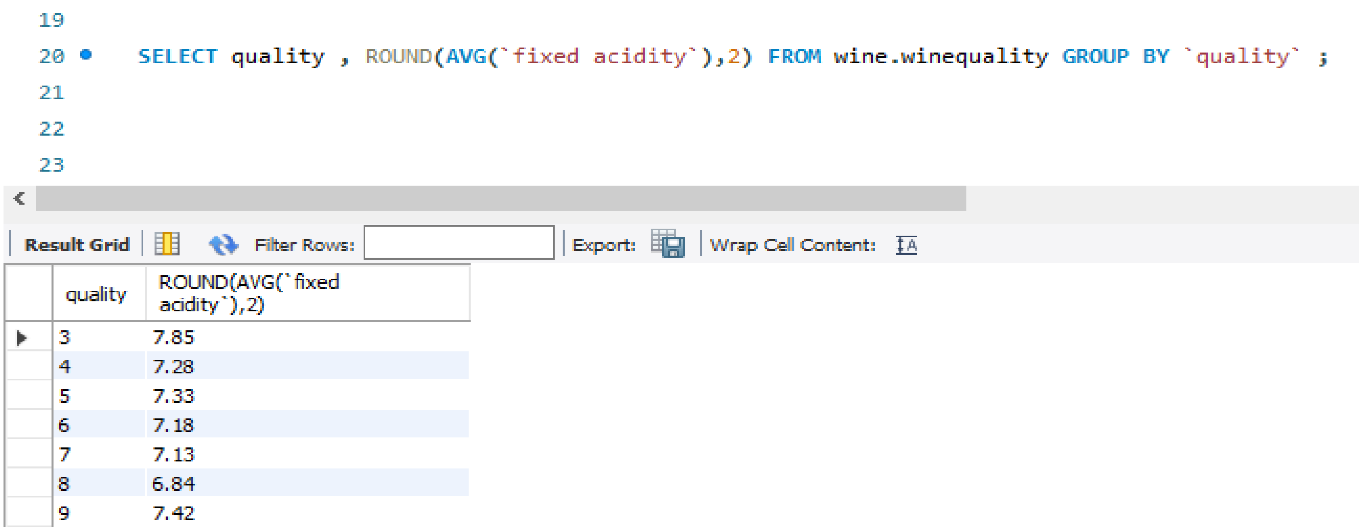
```sql
SELECT type , MIN(`fixed acidity`) AS `MIN FIXED ACIDITY`, MAX(`fixed acidity`) AS `MAX FIXED ACIDITY` FROM wine.winequality
WHERE pH BETWEEN 2.5 AND 7.5 AND `citric acid` < 0.8 GROUP BY `type`;
```

type	MIN FIXED ACIDITY	MAX FIXED ACIDITY
white	3.8	14.2
red	4.6	15.9



```sql
SELECT quality ,COUNT(`total sulfur dioxide`) AS `No. of Products SO2 (70-90)` FROM wine.winequality WHERE `total sulfur dioxide`
BETWEEN 70 AND 90 GROUP BY quality;
```

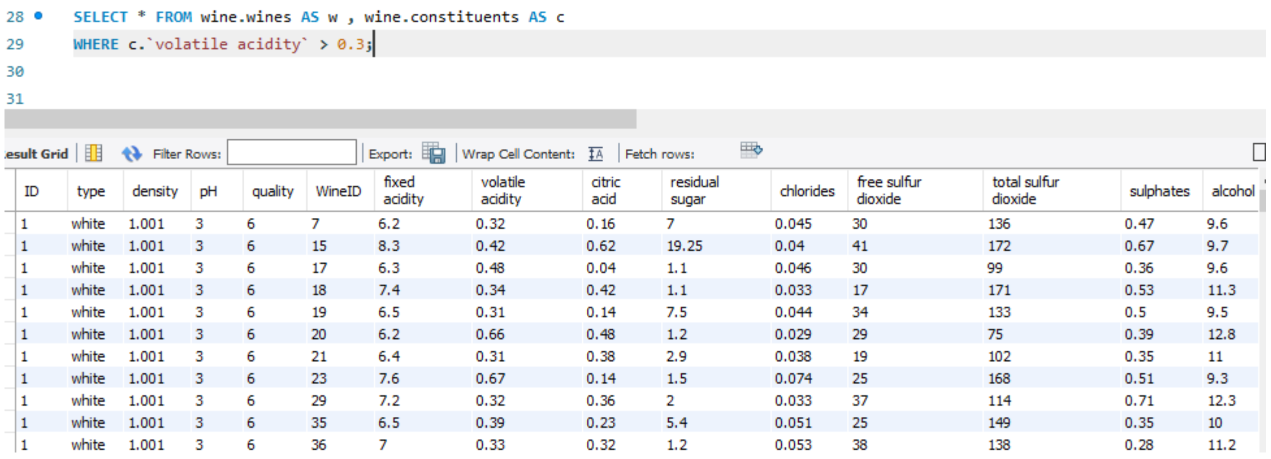
quality	No. of Products SO2 (70-90)
4	23
5	162
6	229
7	114
8	21
9	1



```sql
SELECT quality , ROUND(AVG(`fixed acidity`),2) FROM wine.winequality GROUP BY `quality` ;
```

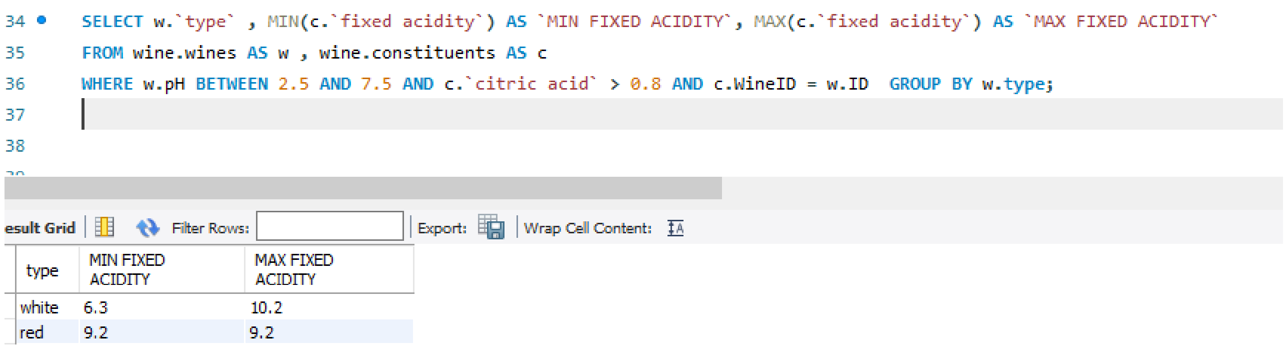
quality	ROUND(AVG(`fixed acidity`),2)
3	7.85
4	7.28
5	7.33
6	7.18
7	7.13
8	6.84
9	7.42

Name: Meith Navlakha
SAP: 60004190068
ADBMS Practical



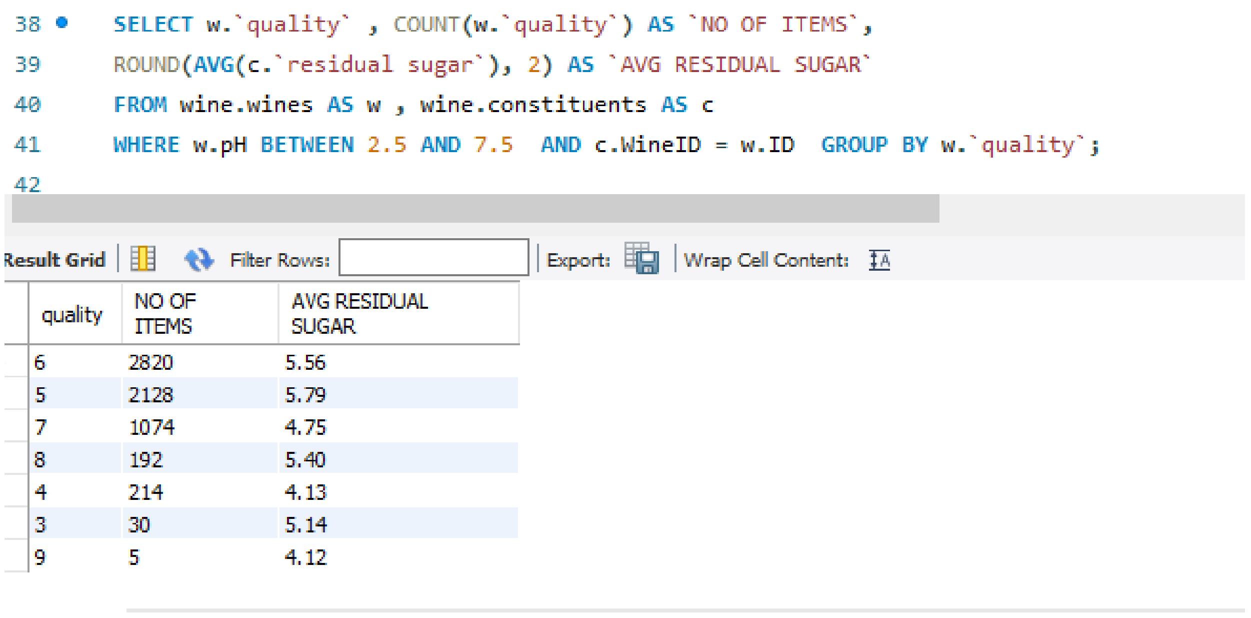
```sql
SELECT * FROM wine.wines AS w , wine.constituents AS c
WHERE c.`volatile acidity` > 0.3;
```

ID	type	density	pH	quality	WineID	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	sulphates	alcohol
1	white	1.001	3	6	7	6.2	0.32	0.16	7	0.045	30	136	0.47	9.6
1	white	1.001	3	6	15	8.3	0.42	0.62	19.25	0.04	41	172	0.67	9.7
1	white	1.001	3	6	17	6.3	0.48	0.04	1.1	0.046	30	99	0.36	9.6
1	white	1.001	3	6	18	7.4	0.34	0.42	1.1	0.033	17	171	0.53	11.3
1	white	1.001	3	6	19	6.5	0.31	0.14	7.5	0.044	34	133	0.5	9.5
1	white	1.001	3	6	20	6.2	0.66	0.48	1.2	0.029	29	75	0.39	12.8
1	white	1.001	3	6	21	6.4	0.31	0.38	2.9	0.038	19	102	0.35	11
1	white	1.001	3	6	23	7.6	0.67	0.14	1.5	0.074	25	168	0.51	9.3
1	white	1.001	3	6	29	7.2	0.32	0.36	2	0.033	37	114	0.71	12.3
1	white	1.001	3	6	35	6.5	0.39	0.23	5.4	0.051	25	149	0.35	10
1	white	1.001	3	6	36	7	0.33	0.32	1.2	0.053	38	138	0.28	11.2



```sql
SELECT w.`type` , MIN(c.`fixed acidity`) AS `MIN FIXED ACIDITY`, MAX(c.`fixed acidity`) AS `MAX FIXED ACIDITY`
FROM wine.wines AS w , wine.constituents AS c
WHERE w.pH BETWEEN 2.5 AND 7.5 AND c.`citric acid` > 0.8 AND c.WineID = w.ID  GROUP BY w.type;
```

type	MIN FIXED ACIDITY	MAX FIXED ACIDITY
white	6.3	10.2
red	9.2	9.2



```sql
SELECT w.`quality` , COUNT(w.`quality`) AS `NO OF ITEMS`,
ROUND(AVG(c.`residual sugar`), 2) AS `AVG RESIDUAL SUGAR`
FROM wine.wines AS w , wine.constituents AS c
WHERE w.pH BETWEEN 2.5 AND 7.5  AND c.WineID = w.ID  GROUP BY w.`quality`;
```

quality	NO OF ITEMS	AVG RESIDUAL SUGAR
6	2820	5.56
5	2128	5.79
7	1074	4.75
8	192	5.40
4	214	4.13
3	30	5.14
9	5	4.12

CONCLUSION: In this experiment, I implemented the basic SQL queries on the Wine Quality dataset. Queries were executed on the original ‘winequality’ dataset as well as the fragments ‘wines’ and ‘constituents’. There were no errors while executing the queries.

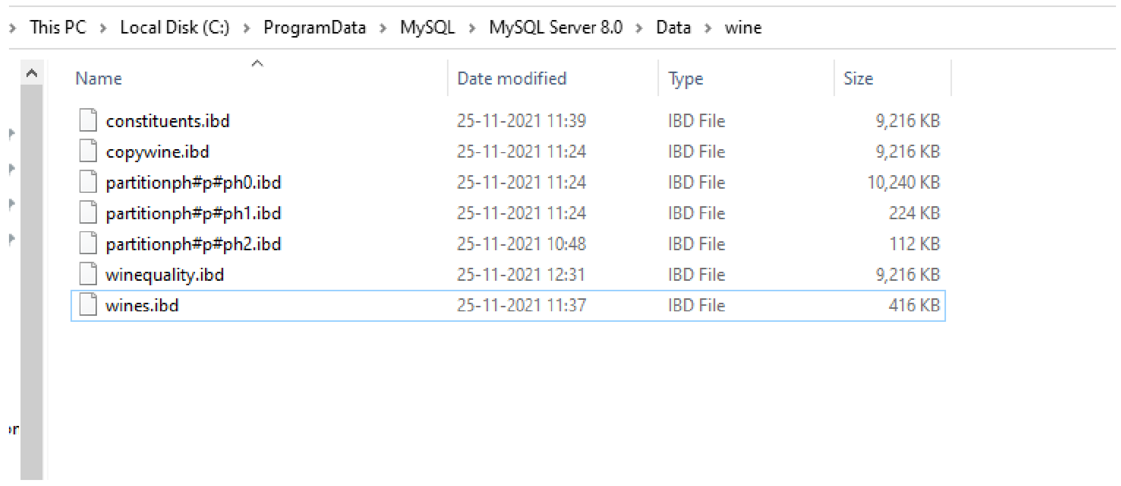
Name: Meith Navlakha
SAP: 60004190068
ADBMS Practical

EXPERIEMNT 6

AIM: Storing of Dataset Fragments of the selected dataset into different drives.

PERFORMANCE:

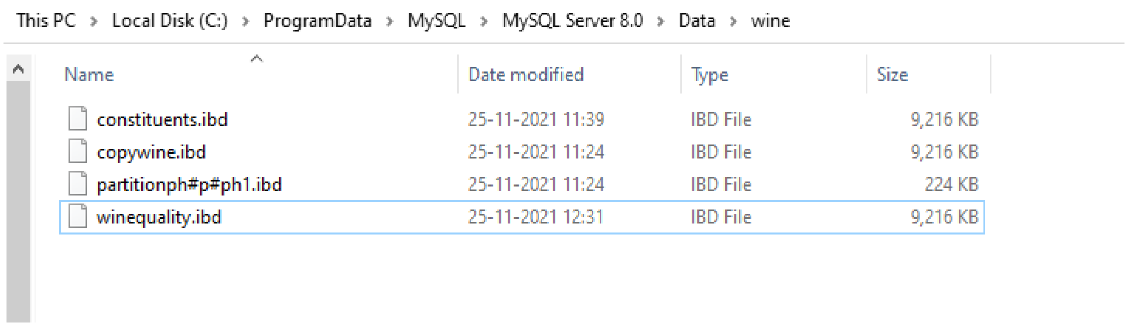
Before distributing the fragments:



This PC › Local Disk (C:) › ProgramData › MySQL › MySQL Server 8.0 › Data › wine

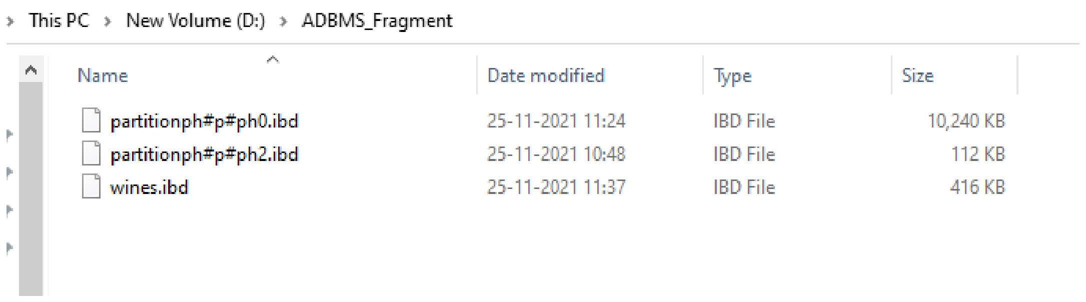
Name	Date modified	Type	Size
constituents.ibd	25-11-2021 11:39	IBD File	9,216 KB
copywine.ibd	25-11-2021 11:24	IBD File	9,216 KB
partitionph#p#ph0.ibd	25-11-2021 11:24	IBD File	10,240 KB
partitionph#p#ph1.ibd	25-11-2021 11:24	IBD File	224 KB
partitionph#p#ph2.ibd	25-11-2021 10:48	IBD File	112 KB
winequality.ibd	25-11-2021 12:31	IBD File	9,216 KB
wines.ibd	25-11-2021 11:37	IBD File	416 KB

After distributing the fragments:



This PC › Local Disk (C:) › ProgramData › MySQL › MySQL Server 8.0 › Data › wine

Name	Date modified	Type	Size
constituents.ibd	25-11-2021 11:39	IBD File	9,216 KB
copywine.ibd	25-11-2021 11:24	IBD File	9,216 KB
partitionph#p#ph1.ibd	25-11-2021 11:24	IBD File	224 KB
winequality.ibd	25-11-2021 12:31	IBD File	9,216 KB



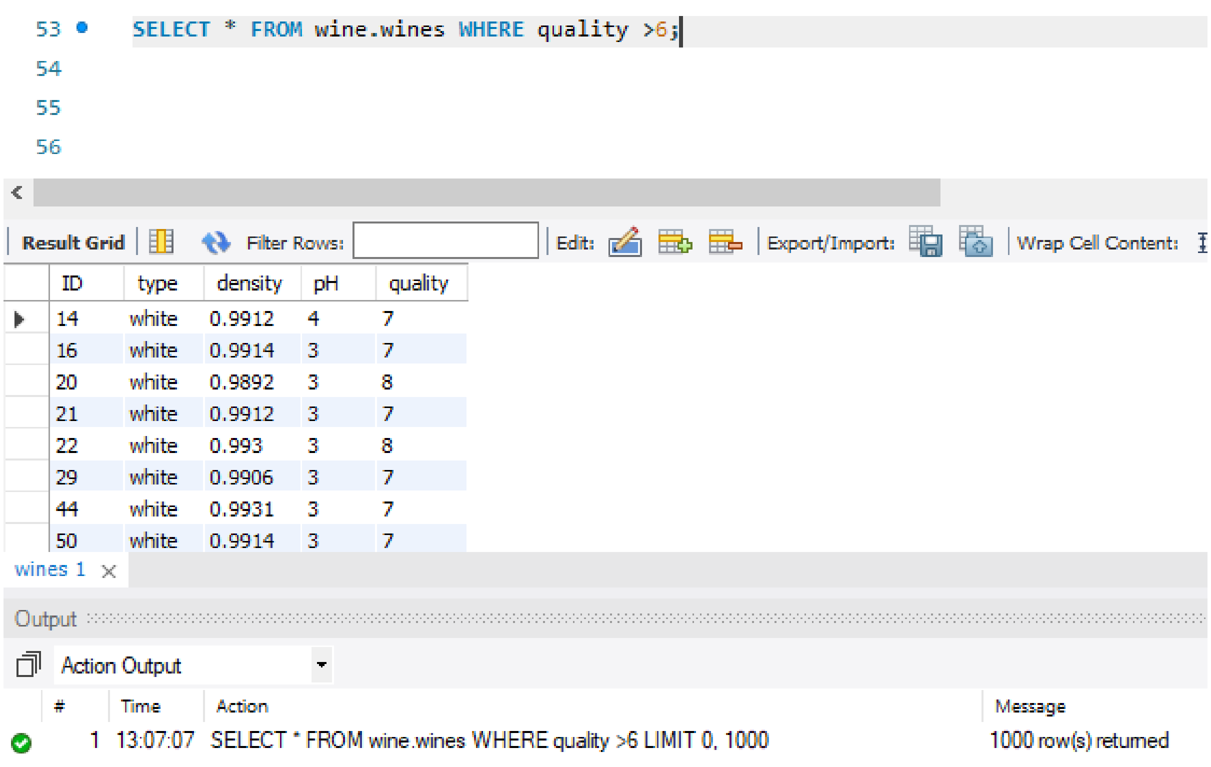
This PC › New Volume (D:) › ADBMS_Fragment

Name	Date modified	Type	Size
partitionph#p#ph0.ibd	25-11-2021 11:24	IBD File	10,240 KB
partitionph#p#ph2.ibd	25-11-2021 10:48	IBD File	112 KB
wines.ibd	25-11-2021 11:37	IBD File	416 KB

Name: Meith Navlakha
SAP: 60004190068
ADBMS Practical

OUTPUT:

Before

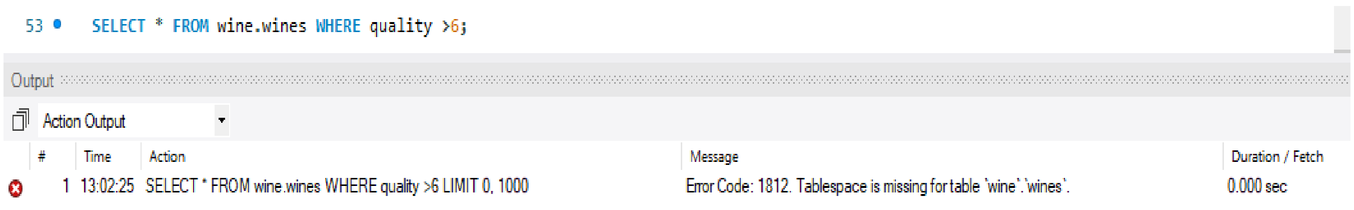


```sql
SELECT * FROM wine.wines WHERE quality >6;
```

ID	type	density	pH	quality
14	white	0.9912	4	7
16	white	0.9914	3	7
20	white	0.9892	3	8
21	white	0.9912	3	7
22	white	0.993	3	8
29	white	0.9906	3	7
44	white	0.9931	3	7
50	white	0.9914	3	7

```
#   Time      Action                                                   Message
1   13:07:07  SELECT * FROM wine.wines WHERE quality >6 LIMIT 0, 1000  1000 row(s) returned
```

After



```sql
SELECT * FROM wine.wines WHERE quality >6;
```

```
#   Time      Action                                                   Message                                                       Duration / Fetch
1   13:02:25  SELECT * FROM wine.wines WHERE quality >6 LIMIT 0, 1000  Error Code: 1812. Tablespace is missing for table `wine`.`wines`.  0.000 sec
```

CONCLUSION: In this experiment, I stored the datasets fragments into different drives but the MySQL workbench is not recognizing any fragments present on the other drives. The error displayed is 1812 i.e. the table is not found(missing).

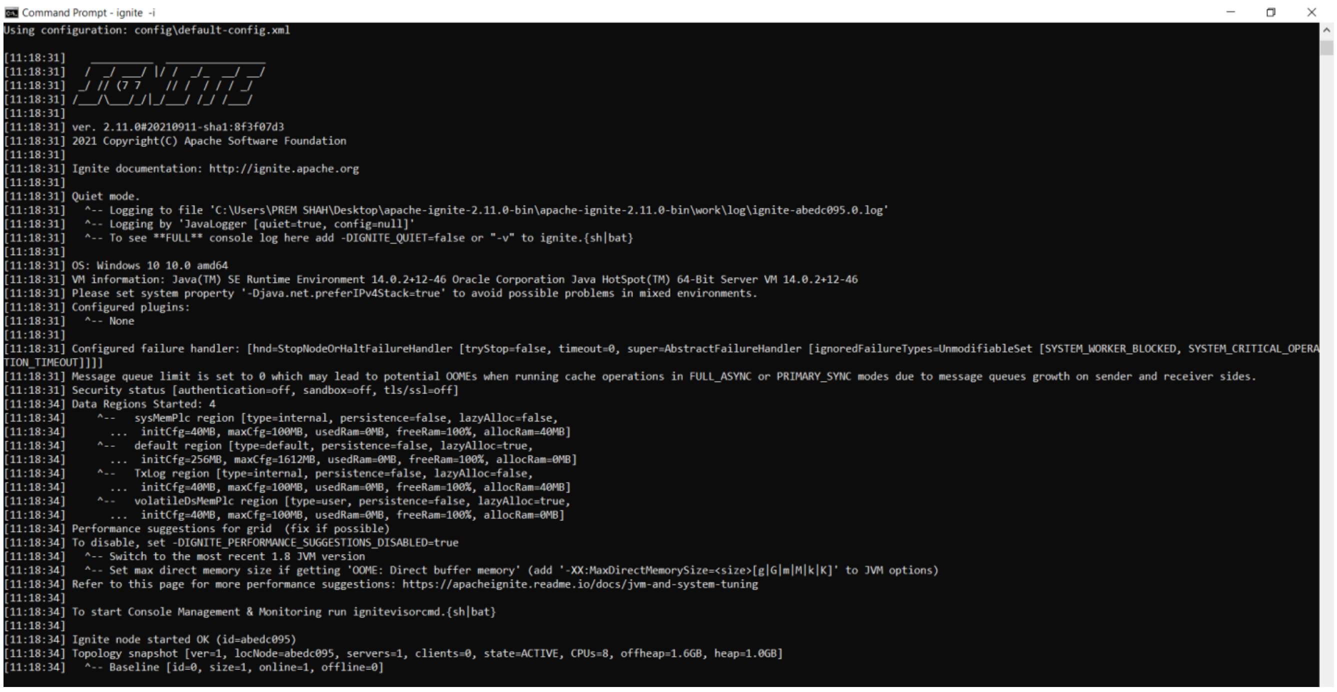
Name: Meith Navlakha
SAP: 60004190068
ADBMS Practical

EXPERIEMNT 7

AIM: Exploring different Distributed Databases Software and Tools and downloading.

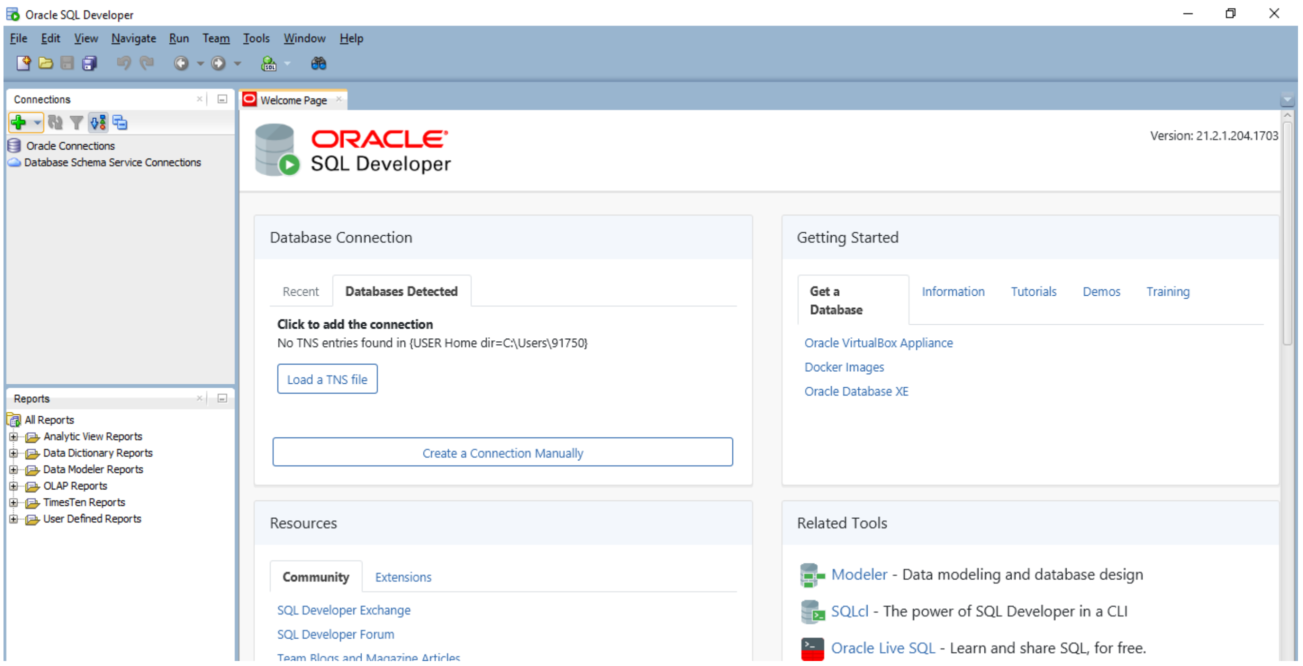
PERFORMANCE:

1) apache-ignite-2.11.0



```
Command Prompt - ignite  -i
Using configuration: config\default-config.xml

[11:18:31]    __________  ________________
[11:18:31]   /  _/ ___/ |/ /  _/_  __/ __/
[11:18:31]  _/ // (7 7    // /  / / / _/
[11:18:31] /___/\___/_/|_/___/ /_/ /___/
[11:18:31]
[11:18:31] ver. 2.11.0#20210911-sha1:8f3f07d3
[11:18:31] 2021 Copyright(C) Apache Software Foundation
[11:18:31]
[11:18:31] Ignite documentation: http://ignite.apache.org
[11:18:31]
[11:18:31] Quiet mode.
[11:18:31]   ^-- Logging to file 'C:\Users\PREM SHAH\Desktop\apache-ignite-2.11.0-bin\apache-ignite-2.11.0-bin\work\log\ignite-abedc095.0.log'
[11:18:31]   ^-- Logging by 'JavaLogger [quiet=true, config=null]'
[11:18:31]   ^-- To see **FULL** console log here add -DIGNITE_QUIET=false or "-v" to ignite.{sh|bat}
[11:18:31]
[11:18:31] OS: Windows 10 10.0 amd64
[11:18:31] VM information: Java(TM) SE Runtime Environment 14.0.2+12-46 Oracle Corporation Java HotSpot(TM) 64-Bit Server VM 14.0.2+12-46
[11:18:31] Please set system property '-Djava.net.preferIPv4Stack=true' to avoid possible problems in mixed environments.
[11:18:31] Configured plugins:
[11:18:31]   ^-- None
[11:18:31]
[11:18:31] Configured failure handler: [hnd=StopNodeOrHaltFailureHandler [tryStop=false, timeout=0, super=AbstractFailureHandler [ignoredFailureTypes=UnmodifiableSet [SYSTEM_WORKER_BLOCKED, SYSTEM_CRITICAL_OPERA
TION_TIMEOUT]]]]
[11:18:31] Message queue limit is set to 0 which may lead to potential OOMEs when running cache operations in FULL_ASYNC or PRIMARY_SYNC modes due to message queues growth on sender and receiver sides.
[11:18:31] Security status [authentication=off, sandbox=off, tls/ssl=off]
[11:18:34] Data Regions Started: 4
[11:18:34]     ^--   sysMemPlc region [type=internal, persistence=false, lazyAlloc=false,
[11:18:34]       ...  initCfg=40MB, maxCfg=100MB, usedRam=0MB, freeRam=100%, allocRam=40MB]
[11:18:34]     ^--   default region [type=default, persistence=false, lazyAlloc=true,
[11:18:34]       ...  initCfg=256MB, maxCfg=1612MB, usedRam=0MB, freeRam=100%, allocRam=0MB]
[11:18:34]     ^--   TxLog region [type=internal, persistence=false, lazyAlloc=false,
[11:18:34]       ...  initCfg=40MB, maxCfg=100MB, usedRam=0MB, freeRam=100%, allocRam=40MB]
[11:18:34]     ^--   volatileDsMemPlc region [type=user, persistence=false, lazyAlloc=true,
[11:18:34]       ...  initCfg=40MB, maxCfg=100MB, usedRam=0MB, freeRam=100%, allocRam=0MB]
[11:18:34] Performance suggestions for grid  (fix if possible)
[11:18:34] To disable, set -DIGNITE_PERFORMANCE_SUGGESTIONS_DISABLED=true
[11:18:34]   ^-- Switch to the most recent 1.8 JVM version
[11:18:34]   ^-- Set max direct memory size if getting 'OOME: Direct buffer memory' (add '-XX:MaxDirectMemorySize=<size>[g|G|m|M|k|K]' to JVM options)
[11:18:34] Refer to this page for more performance suggestions: https://apacheignite.readme.io/docs/jvm-and-system-tuning
[11:18:34]
[11:18:34] To start Console Management & Monitoring run ignitevisorcmd.{sh|bat}
[11:18:34]
[11:18:34] Ignite node started OK (id=abedc095)
[11:18:34] Topology snapshot [ver=1, locNode=abedc095, servers=1, clients=0, state=ACTIVE, CPUs=8, offheap=1.6GB, heap=1.0GB]
[11:18:34]   ^-- Baseline [id=0, size=1, online=1, offline=0]
```

2) Oracle sql developer



Click to add the connection
No TNS entries found in {USER Home dir=C:\Users\91750}

Modeler - Data modeling and database design
SQLcl - The power of SQL Developer in a CLI
Oracle Live SQL - Learn and share SQL, for free.

Name: Meith Navlakha
SAP: 60004190068
ADBMS Practical

Apache Ignite:

Apache Ignite is a distributed database for high-performance computing with inmemory speed. Apache Ignite's database utilizes RAM as the default storage and processing tier, thus, belonging to the class of in-memory computing platforms. Regardless of the API used, data in Ignite is stored in the form of key- value pairs. The database component scales horizontally, distributing key-value pairs across the cluster in such a way that every node owns a portion of the overall data set. Data is rebalanced automatically whenever a node is added to or removed from the cluster. On top of its distributed foundation, Apache Ignite supports a variety of APIs including JCache-compliant key-value APIs, ANSI-99 SQL with joins, ACID transactions, as well as MapReduce like computations.

Apache Ignite cluster can be deployed on-premise on a commodity hardware, in the cloud (e.g. Microsoft Azure, AWS, Google Compute Engine) or in a containerized and provisioning environments such as Kubernetes, Docker, Apache Mesos, VMWare.

CONCLUSION: In this experiment, I explored through different Distributed Databases software and finally downloaded apache-ignite-2.11.0 and Oracle sql developer for the upcoming experiments for performing distributed databases concepts. I will be using Apache Ignite for my Distributed Databases experiments.