Query Report

```
sql1 = "SELECT users.id, users.name, COALESCE (SUM(o.price), 0) AS amount, COUNT(*)
OVER () as totalRows" + " FROM (SELECT name, id FROM users ORDER BY name ASC
OFFSET "+row_offset+" )" +
                                        " AS users LEFT JOIN orders o" +
                                        " ON (o.user_id = users.id)" +
                                        " GROUP BY users.id, users.name" +
                                        " ORDER BY users.name asc LIMIT 20"          ;
```

Small Database
Before Indices Query 1 time = 186 ms
After Indices Query 1 time = 37 ms

Large Database
Before Indices Query 1 time = 164210 ms
After Indices Query 1 time = 37859 ms

```
sql1 = "SELECT users.id, users.name, COALESCE (SUM(o.price), 0) AS amount, COUNT(*)
OVER () as totalRows" + " FROM (SELECT name, id FROM users ORDER BY name ASC
OFFSET "+row_offset+" )" +
                                        " AS users LEFT JOIN orders o" +
                                        " ON (o.user_id = users.id)" +
                                        " GROUP BY users.id, users.name" +
                                        " order by COALESCE (SUM(o.price), 0)  desc LIMIT 20"
        ;
```

Small Database
Before Indices Query 1 time = 179 ms
After Indices Query 1 time = 36 ms

Large Database
Before Indices Query 1 time = 174593 ms
After Indices Query 1 time = 32964 ms

```
sql1 = "SELECT state.state, COALESCE (SUM(o.price), 0) AS amount, COUNT(*) OVER () as
totalRows" + " FROM (SELECT state, id FROM users ORDER BY state ASC OFFSET
"+row_offset+")" +
                                        " AS state LEFT JOIN orders o" +
                                        " ON (o.user_id = state.id)" +
                                        " GROUP BY state.state" +
                                        " order by state.state  asc LIMIT 20"
```

Small Database
Before Indices Query 1 time = 184 ms
After Indices Query 1 time = 35 ms

Large Database
Before Indices Query 1 time = 163844 ms
After Indices Query 1 time = 45023 ms

```
sql1 = "SELECT state.state, COALESCE (SUM(o.price), 0) AS amount, COUNT(*) OVER () as
totalRows" + " FROM (SELECT state, id FROM users ORDER BY state ASC OFFSET
"+row_offset+")" +
                                        " AS state LEFT JOIN orders o" +
                                        " ON (o.user_id = state.id)" +
                                        " GROUP BY state.state" +
                 " order by COALESCE (SUM(o.price), 0)  desc LIMIT 20"
```

Small Database
Before Indices Query 1 time = 197 ms
After Indices Query 1 time = 29 ms

Large Database
Before Indices Query 1 time = 154978 ms
After Indices Query 1 time = 43045 ms

```
sql2= "SELECT prod.id, prod.name, COALESCE (SUM(orders.price), 0) AS amount, COUNT(*)
OVER () as totalCols" + " FROM (SELECT id, name FROM products ORDER BY name ASC
OFFSET "+col_offset+ ") AS prod" +
                    " LEFT JOIN orders ON (orders.product_id = prod.id)" +
                                        " GROUP BY prod.id, prod.name" +
                                        " order by prod.name asc LIMIT 10";
```

Small Database
Before Indices Query 2 time = 25 ms
After Indices Query 2 time = 5 ms

Large Database
Before Indices Query 2 time = 23.3 minutes
After Indices Query 2 time = 15.3 minutes

```
sql2 = "SELECT prod.id, prod.name, COALESCE (SUM(orders.price), 0) AS amount, COUNT(*)
OVER () as totalCols" + " FROM (SELECT id, name FROM products where
products.category_id = " +state+
                              " ORDER BY name ASC OFFSET "+col_offset+ ") AS prod" +
                                        " LEFT JOIN orders ON (orders.product_id = prod.id)" +
                                        " GROUP BY prod.id, prod.name" +
                                        " order by prod.name asc LIMIT 10";
```

Small Database
Before Indices Query 2 time = 27 ms
After Indices Query 2 time = 6 ms

Large Database
Before Indices Query 2 time = 22.9 minutes
After Indices Query 2 time = 16.4 minutes

```
sql2 = "SELECT prod.id, prod.name, COALESCE (SUM(orders.price), 0) AS amount, COUNT(*)
OVER () as totalCols" + " FROM (SELECT id, name FROM products ORDER BY name ASC
OFFSET "+col_offset+ ") AS prod" +
                                        " LEFT JOIN orders ON (orders.product_id =
                                prod.id)" + " GROUP BY prod.id, prod.name"
                        + " order by COALESCE (SUM(orders.price), 0) desc LIMIT 10”;
```

Small Database
Before Indices Query 2 time = 24 ms
After Indices Query 2 time = 5 ms

Large Database
Before Indices Query 2 time = 23.6 minutes
After Indices Query 2 time = 10.7 minutes

```
sql2 = "SELECT prod.id, prod.name, COALESCE (SUM(orders.price), 0) AS amount, COUNT(*)
OVER () as totalCols" + " FROM (SELECT id, name FROM products where
products.category_id = " +state+
             " ORDER BY name ASC OFFSET "+col_offset+ ") AS prod" + "
LEFT JOIN orders ON (orders.product_id = prod.id)" +
                " GROUP BY prod.id, prod.name" +
                " order by COALESCE (SUM(orders.price), 0) desc LIMIT 10 ";
```
Small Database
Before Indices Query 2 time = 29 ms
After Indices Query 2 time = 7 ms

Large Database
Before Indices Query 2 time = 25.3 minutes
After Indices Query 2 time = 12.6 minutes

```
sql3 = "SELECT prod.id, prod.name, COALESCE (SUM(orders.price), 0) AS amount" + " FROM
(SELECT id, name FROM products LIMIT 10 OFFSET "+col_offset+ " ) AS prod" + " LEFT JOIN
orders ON (orders.product_id = prod.id and orders.user_id =" +Integer.toString(r1.getInt("id"))+
")" +
                                                        " GROUP BY prod.id, prod.name" +
                                                        " order by prod.name asc”;
```

Small Database
Before Indices Query 3 time = 3.5 ms
After Indices Query 3 time = 1.5 ms

Large Database
Before Indices Query 3 time = 4345 ms
After Indices Query 3 time = 289 ms

```
sql3 = "SELECT prod.id, prod.name, COALESCE (SUM(orders.price), 0) AS amount" + " FROM
(SELECT id, name FROM products WHERE products.category_id = " +state+  " LIMIT 10
OFFSET "+col_offset+ " ) AS prod" + " LEFT JOIN orders ON (orders.product_id = prod.id and
orders.user_id =" +Integer.toString(r1.getInt("id"))+ ")" +
                                                        " GROUP BY prod.id, prod.name" +
                                                        " order by prod.name asc”;
```

Small Database
Before Indices Query 3 time = 2.8 ms
After Index Query 3 time = 1.4 ms

Large Database
Before Indices Query 3 time = 4297 ms
After Indices Query 3 time = 321 ms

```
sql3 = "SELECT prod.id, prod.name, COALESCE (SUM(orders.price), 0) AS amount" + " FROM
(SELECT id, name FROM products LIMIT 10 OFFSET "+col_offset+ ") AS prod" +           "
LEFT JOIN orders ON (orders.product_id = prod.id and orders.user_id ="
+Integer.toString(r1.getInt("id"))+ ")" +  " GROUP BY prod.id, prod.name" +
                              " order by COALESCE (SUM(orders.price), 0)  DESC ";
```

Small Database
Before Indices Query 3 time = 3.8 ms
After Index Query 3 time = 1.6 ms

Large Database
Before Indices Query 3 time = 3976 ms
After Indices Query 3 time = 314 ms

```
sql3 = "SELECT prod.id, prod.name, COALESCE (SUM(orders.price), 0) AS amount" + " FROM
(SELECT id, name FROM products WHERE products.category_id = " +state+  " LIMIT 10
OFFSET "+col_offset+ ") AS prod" + " LEFT JOIN orders ON (orders.product_id = prod.id and
orders.user_id =" +Integer.toString(r1.getInt("id"))+ ")" +
" GROUP BY prod.id, prod.name" +
" order by COALESCE (SUM(orders.price), 0) DESC          ";
```

Small Database
Before Indices Query 3 time = 3.2 ms
After Index Query 3 time = 1.5 ms

Large Database
Before Indices Query 3 time = 4195 ms
After Indices Query 3 time = 267 ms

```
sql3 =   "SELECT prod.id, prod.name, COALESCE (SUM(orders.price), 0) AS amount" + "
FROM (SELECT id, name FROM products LIMIT 10 OFFSET "+col_offset+ ") AS prod "
+         " LEFT JOIN orders ON (orders.product_id = prod.id " +      " and orders.user_id in
(select id from users where state = "'" +rowName+  "' ) )" +
                                 " GROUP BY prod.id, prod.name " +
                                " order by prod.name asc" ;
```

Small Database
Before Indices Query 3 time = 4.1 ms
After Index Query 3 time = 1.4 ms

Large Database
Before Indices Query 3 time = 4694 ms
After Indices Query 3 time = 512 ms

```
sql3 =   "SELECT prod.id, prod.name, COALESCE (SUM(orders.price), 0) AS amount" + "
FROM (SELECT id, name FROM products WHERE products.category_id = " +state+  " LIMIT
10 OFFSET "+col_offset+ ") AS prod " + " LEFT JOIN orders ON (orders.product_id = prod.id "
+
" and orders.user_id in (select id from users where state = "'" +rowName+  "' ) )" + " GROUP BY
prod.id, prod.name " +
                                            " order by prod.name asc" ;
```

Small Database
Before Indices Query 3 time = 3.6 ms
After Index Query 3 time = 1.5 ms

Large Database
Before Indices Query 3 time = 4496 ms
After Indices Query 3 time = 493 ms

```
sql3 =   "SELECT prod.id, prod.name, COALESCE (SUM(orders.price), 0) AS amount" + " FROM (SELECT id, name FROM products LIMIT 10 OFFSET "+col_offset+ ") AS prod " + " LEFT JOIN orders ON (orders.product_id = prod.id " + " and orders.user_id in (select id from users where state = '" +rowName+  "' ) )" +
                              " GROUP BY prod.id, prod.name " +
                              " order by COALESCE (SUM(orders.price), 0)  DESC" ;
```

Small Database
Before Indices Query 3 time = 3.7 ms
After Index Query 3 time = 1.3 ms

Large Database
Before Indices Query 3 time = 5012 ms
After Indices Query 3 time = 468 ms

```
sql3 =   "SELECT prod.id, prod.name, COALESCE (SUM(orders.price), 0) AS amount" + " FROM (SELECT id, name FROM products WHERE products.category_id = " +state+  " LIMIT 10 OFFSET "+col_offset+ ") AS prod " + " LEFT JOIN orders ON (orders.product_id = prod.id " + ” and orders.user_id in (select id from users where state = '" +rowName+  "' ) )" +
                              ” GROUP BY prod.id, prod.name " +
                              " order by COALESCE (SUM(orders.price), 0)  DESC" ;
```

Small Database
Before Indices Query 3 time = 2.9 ms
After Index Query 3 time = 1.4 ms

Large Database
Before Indices Query 3 time = 4963 ms
After Indices Query 3 time = 296 ms

```
Similar_product_SQL = "SELECT p1.id AS p1id, p2.id AS p2id, (COALESCE((SELECT
SUM(ord1.price * ord2.price)" +
                                        " FROM orders ord1, orders ord2" +
                                        " WHERE ord1.product_id = p1.id AND ord2.product_id = p2.id
AND ord1.user_id = ord2.user_id),0)) /" +
                                        " (SQRT ((SELECT SUM(POWER(price,2)) FROM orders
WHERE product_id = p1.id)) * SQRT ((SELECT SUM(POWER(price,2)) FROM orders WHERE
product_id = p2.id))) AS cosine" +
                              " FROM  products p1, products p2" +
                              " WHERE p1.id < p2.id" +
                              " AND p1.id IN (Select product_id FROM orders)" +
                              " AND p2.id IN (Select product_id FROM orders)" +
                              " GROUP BY p1id,p2id" +
                              " ORDER BY cosine DESC" ;
```

Small Database
Before Indices Query 3 time = 5.2 minutes
After Index Query 3 time = 1.3 minutes

Large Database
Before Indices Query 3 time = >30 minutes
After Indices Query 3 time = > 30 minutes

<u>Index Report</u>

Beneficial Indexes (final choices):

create index user_index on users(name, state);
This index significantly sped up our “sql1”. Our “sql1” query gets the row headers, whether its users or states, so the index makes getting the user names or the state names from the users incredibly fast. The select operation is seamless because the names/states of the users has been indexed.

create index order_index on orders(user_id, product_id);
This index is very beneficial because it allows for quick querying and searching of orders of their product id. For our “sql3” query, we do a left join with products and orders and match the tables with the orders product_id and the user_id of the user being looked at. This index increased the speed by up to 10x depending on the size of the database.
This index also became very very beneficial to have with the similar products page query. The sequential access for the query for the similar products would have to scan all the orders and match them with products, but with the index, the scanning took far less time, which put less load on the CPU of the computer.

<u>Experimental Results and Analysis Not Beneficial indexes</u>

create index prod_index on products(name, price, category_id);

This index did not improve running time on any of the queries. We believe this is because we are issuing a select command on the products before we do a left join with the orders table. It is faster to do a sequential access for this type of command instead of using the index.

create index user_id_index on users(id);

This index did not improve the running time of any of the queries. The reason we believe is because we are never doing a select for the users id. Our queries match a specific user’s id from the order’s table, but the user id is custom and already been pre set from the result of a previous query. So this index is never really used for its purpose.

CREATE INDEX idx_order_user_id on orders(user_id);
CREATE INDEX idx_order_user_id on orders(user_id);
We tried creating and using both these index’s for the similar products page and the speed up was not that great. W believe that indexes won’t really be as much use in looking up similar

products because the products, orders, and users tables will constantly be updated with new users, products, or orders and hence all those columns will be constantly manipulated. Without or without the indexes the query time was greater than 30 mins.