Based on the dataset chosen for Project 1, evaluate the performance of queries before and after adding new indexes by timing them. Also, exploring a new dataset on MongoDB.

Part 1: Retail data of a Coffee Chain
Part 2: Bikez dataset

Project 2 – Database Foundations for Business Analytics (BUAN 6320.005)

Group 11:
Maneka Prabhu (MXP210102)
Rishapp Rajesh (RXR210108)
Shivanag Avula (SXA162630)
Venkata Durga Sai Kowshik Vooda (VXV210032)
Vikhil Baswaraju (VXB210053)

Part 1: Indexing and Query Timing

The dataset we chose contains representative retail data for a coffee chain.

1.1. Below is a list all the current indexes in the database table wise:

1. Customer table

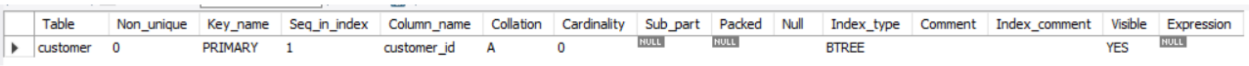


Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
customer	0	PRIMARY	1	customer_id	A	0	NULL	NULL		BTREE			YES	NULL

2. Staff Table

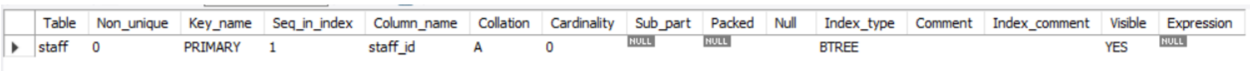


Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
staff	0	PRIMARY	1	staff_id	A	0	NULL	NULL		BTREE			YES	NULL

3. Dates Table

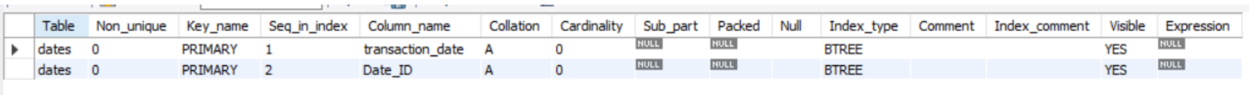


Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
dates	0	PRIMARY	1	transaction_date	A	0	NULL	NULL		BTREE			YES	NULL
dates	0	PRIMARY	2	Date_ID	A	0	NULL	NULL		BTREE			YES	NULL

4. Generations Table

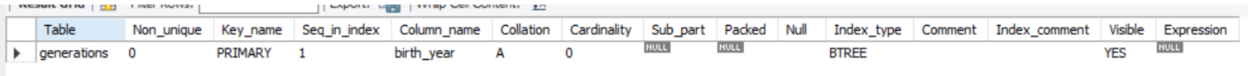


Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
generations	0	PRIMARY	1	birth_year	A	0	NULL	NULL		BTREE			YES	NULL

5. Product Table

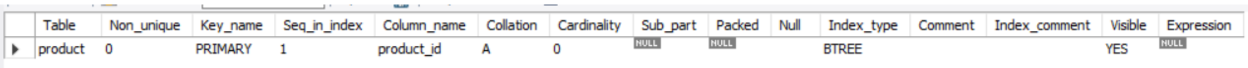


Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
product	0	PRIMARY	1	product_id	A	0	NULL	NULL		BTREE			YES	NULL

6. Pastry Inventory Table

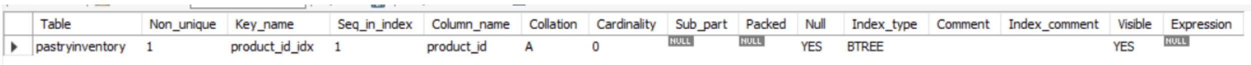


Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
pastryinventory	1	product_id_idx	1	product_id	A	0	NULL	NULL	YES	BTREE			YES	NULL

7. Sales Outlet Table

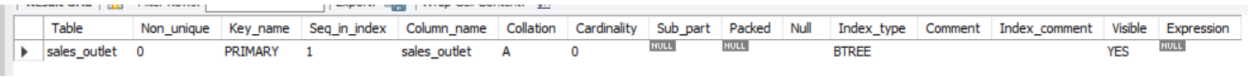


Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
sales_outlet	0	PRIMARY	1	sales_outlet	A	0	NULL	NULL		BTREE			YES	NULL

8. Sales Receipts Table

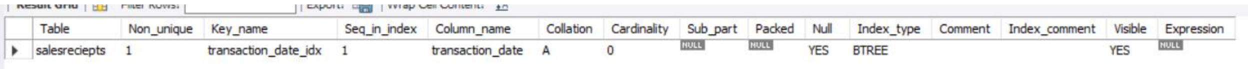


Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
salesreciepts	1	transaction_date_idx	1	transaction_date	A	0	NULL	NULL	YES	BTREE			YES	NULL

9. Sales Targets Table

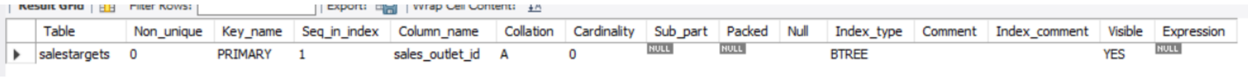
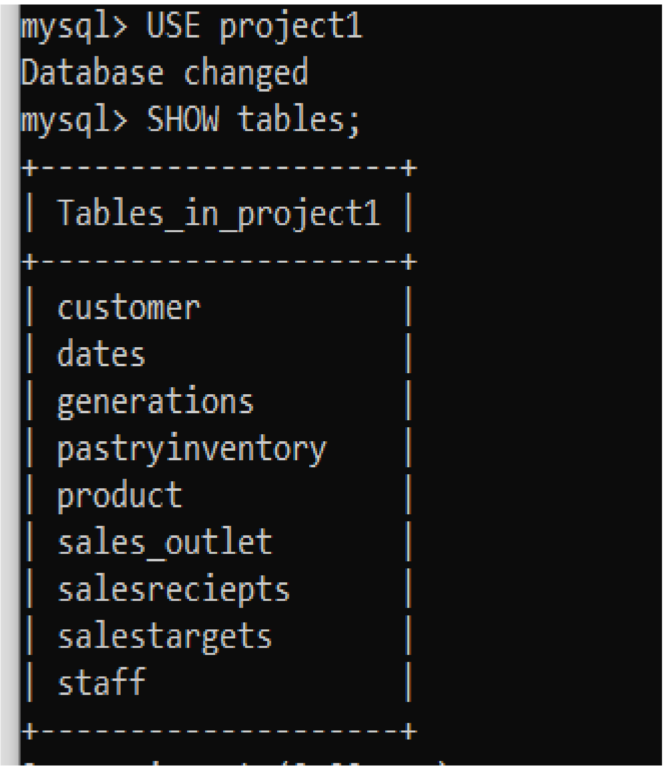


Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
salestargets	0	PRIMARY	1	sales_outlet_id	A	0	NULL	NULL		BTREE			YES	NULL

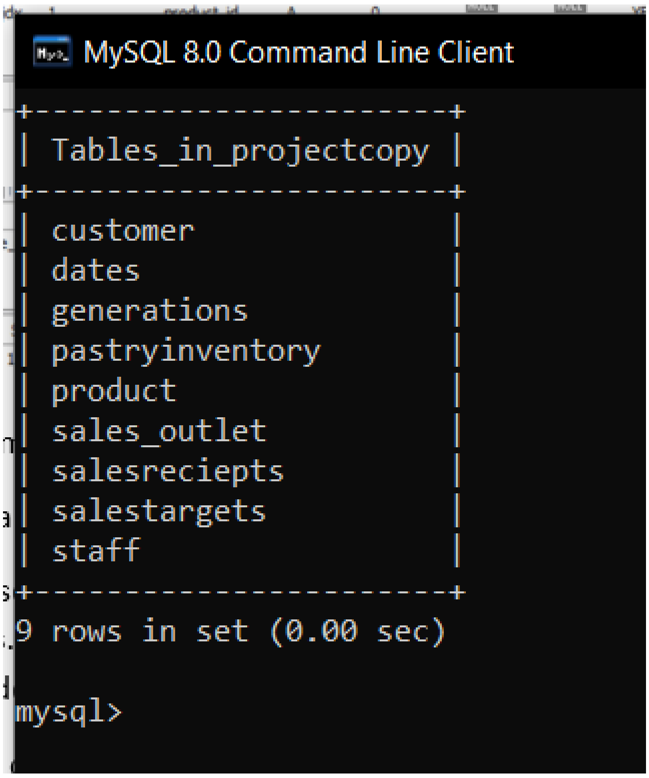
1.2. Common factors in these columns:

It seems that the database management system indexes the primary key and the common keys, the foreign keys in the tables. The Primary Keys are indexed as primary whereas the foreign keys selected have actual key name indexes.

1.3. Made copy of database through MySQL dump command and deleted indexes through the index tab in MySQL Workbench.



```
mysql> USE project1
Database changed
mysql> SHOW tables;
+--------------------+
| Tables_in_project1 |
+--------------------+
| customer           |
| dates              |
| generations        |
| pastryinventory    |
| product            |
| sales_outlet       |
| salesreciepts      |
| salestargets       |
| staff              |
+--------------------+
```



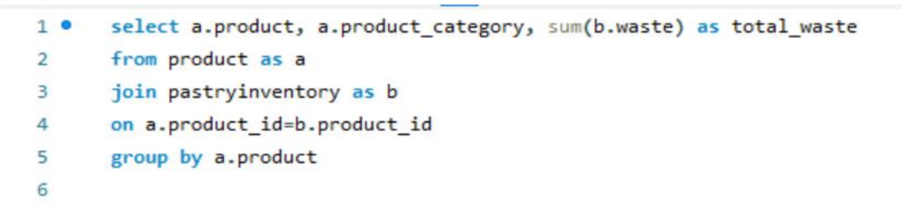
MySQL 8.0 Command Line Client

```
+-----------------------+
| Tables_in_projectcopy |
+-----------------------+
| customer              |
| dates                 |
| generations           |
| pastryinventory       |
| product               |
| sales_outlet          |
| salesreciepts         |
| salestargets          |
| staff                 |
+-----------------------+
9 rows in set (0.00 sec)

mysql>
```

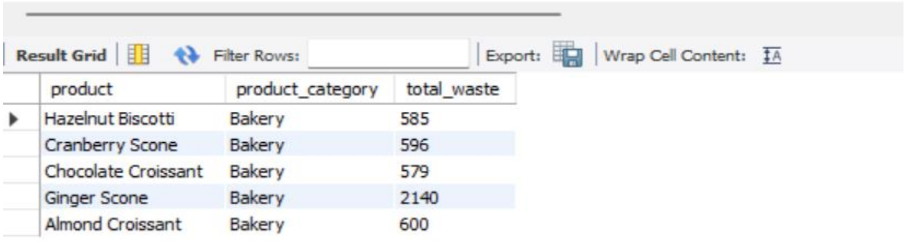
1.4. 5 queries using JOINS

1. Query to list the products and the inventory wasted for each product:

   select a.product, a.product_category, sum(b.waste) as total_waste
   from product as a
   join pastryinventory as b
   on a.product_id=b.product_id
   group by a.product



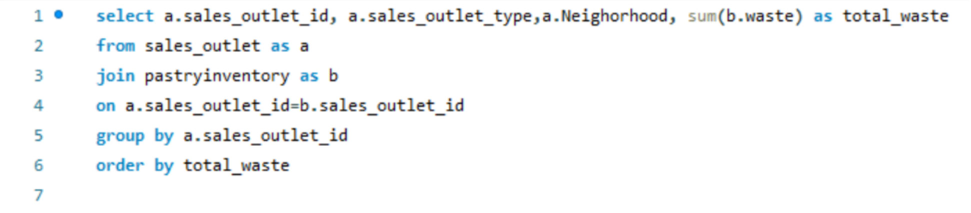
```sql
select a.product, a.product_category, sum(b.waste) as total_waste
from product as a
join pastryinventory as b
on a.product_id=b.product_id
group by a.product

```



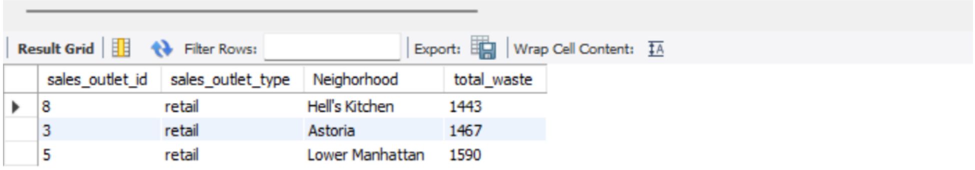
product	product_category	total_waste
Hazelnut Biscotti	Bakery	585
Cranberry Scone	Bakery	596
Chocolate Croissant	Bakery	579
Ginger Scone	Bakery	2140
Almond Croissant	Bakery	600

2. Query to list the sales outlets by increasing amount of inventory wastage

   select a.sales_outlet_id, a.sales_outlet_type,a.Neighorhood, sum(b.waste) as total_waste
   from sales_outlet as a
   join pastryinventory as b
   on a.sales_outlet_id=b.sales_outlet_id
   group by a.sales_outlet_id
   order by total_waste



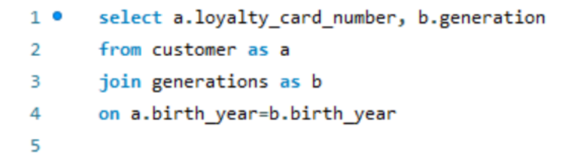
```
select a.sales_outlet_id, a.sales_outlet_type,a.Neighorhood, sum(b.waste) as total_waste
from sales_outlet as a
join pastryinventory as b
on a.sales_outlet_id=b.sales_outlet_id
group by a.sales_outlet_id
order by total_waste
```



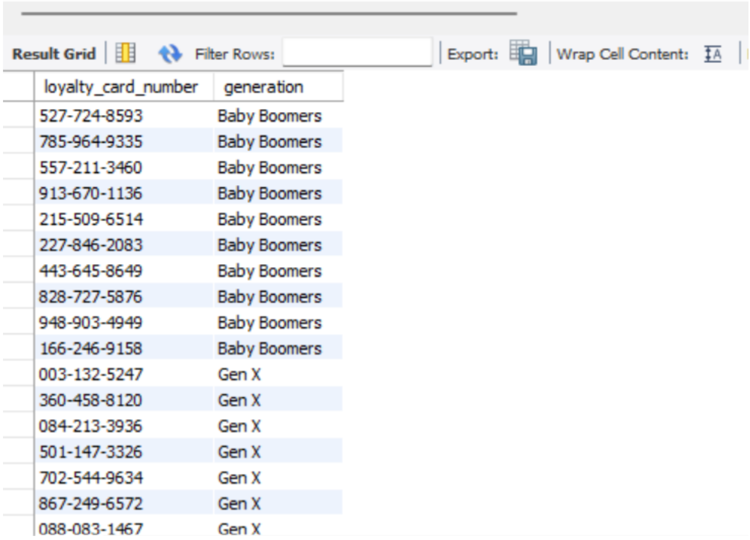
sales_outlet_id	sales_outlet_type	Neighorhood	total_waste
8	retail	Hell's Kitchen	1443
3	retail	Astoria	1467
5	retail	Lower Manhattan	1590

3. Query to list customers based on loyalty numbers and generation they belong to

   select a.loyalty_card_number, b.generation
   from customer as a
   join generations as b
   on a.birth_year=b.birth_year



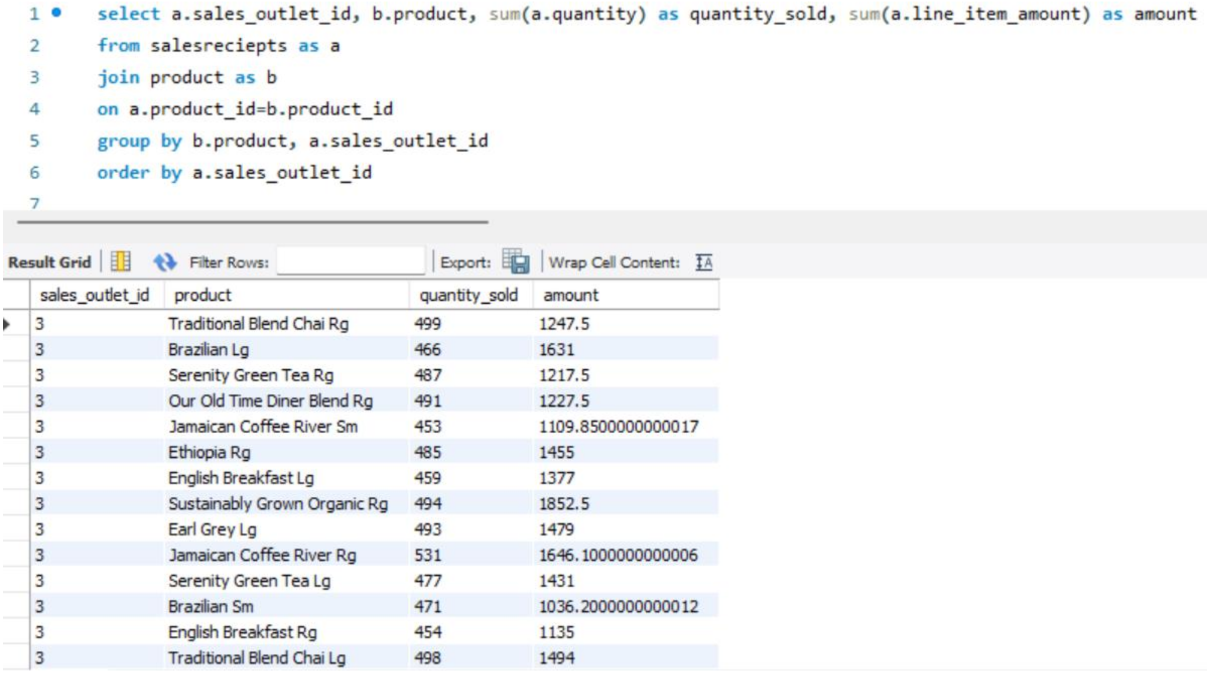
```
select a.loyalty_card_number, b.generation
from customer as a
join generations as b
on a.birth_year=b.birth_year
```



loyalty_card_number	generation
527-724-8593	Baby Boomers
785-964-9335	Baby Boomers
557-211-3460	Baby Boomers
913-670-1136	Baby Boomers
215-509-6514	Baby Boomers
227-846-2083	Baby Boomers
443-645-8649	Baby Boomers
828-727-5876	Baby Boomers
948-903-4949	Baby Boomers
166-246-9158	Baby Boomers
003-132-5247	Gen X
360-458-8120	Gen X
084-213-3936	Gen X
501-147-3326	Gen X
702-544-9634	Gen X
867-249-6572	Gen X
088-083-1467	Gen X

4. Query to list the amount earned from each product in each store

```
select a.sales_outlet_id, b.product, sum(a.quantity) as quantity_sold,
sum(a.line_item_amount) as amount
from salesreciepts as a
join product as b
on a.product_id=b.product_id
group by b.product, a.sales_outlet_id
order by a.sales_outlet_id
```

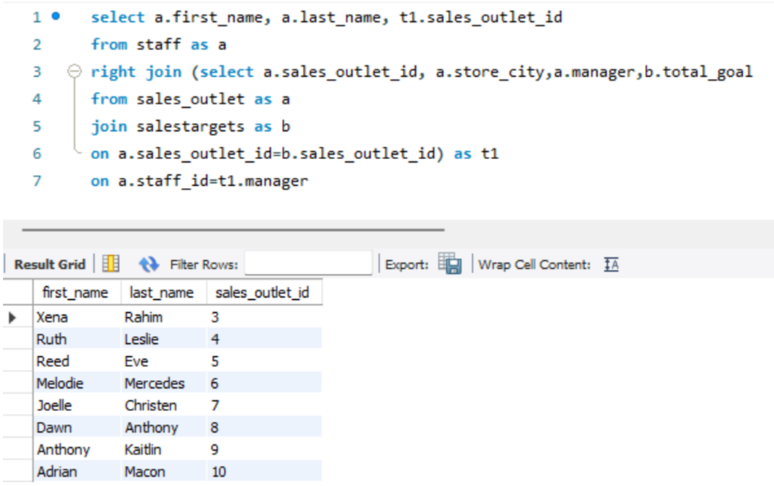


```
select a.sales_outlet_id, b.product, sum(a.quantity) as quantity_sold, sum(a.line_item_amount) as amount
from salesreciepts as a
join product as b
on a.product_id=b.product_id
group by b.product, a.sales_outlet_id
order by a.sales_outlet_id
```

sales_outlet_id	product	quantity_sold	amount
3	Traditional Blend Chai Rg	499	1247.5
3	Brazilian Lg	466	1631
3	Serenity Green Tea Rg	487	1217.5
3	Our Old Time Diner Blend Rg	491	1227.5
3	Jamaican Coffee River Sm	453	1109.8500000000017
3	Ethiopia Rg	485	1455
3	English Breakfast Lg	459	1377
3	Sustainably Grown Organic Rg	494	1852.5
3	Earl Grey Lg	493	1479
3	Jamaican Coffee River Rg	531	1646.1000000000006
3	Serenity Green Tea Lg	477	1431
3	Brazilian Sm	471	1036.2000000000012
3	English Breakfast Rg	454	1135
3	Traditional Blend Chai Lg	498	1494

5. Query returning manager names for each store

```
select a.first_name, a.last_name, t1.sales_outlet_id
from staff as a
right join (select a.sales_outlet_id, a.store_city,a.manager,b.total_goal
from sales_outlet as a
join salestargets as b
on a.sales_outlet_id=b.sales_outlet_id) as t1
on a.staff_id=t1.manager
```

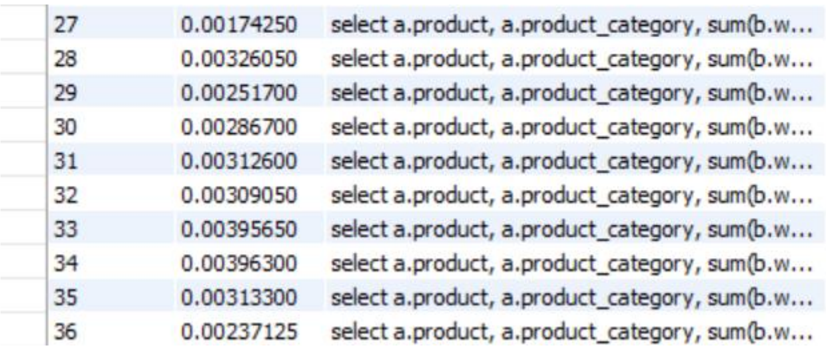


```sql
select a.first_name, a.last_name, t1.sales_outlet_id
from staff as a
right join (select a.sales_outlet_id, a.store_city,a.manager,b.total_goal
from sales_outlet as a
join salestargets as b
on a.sales_outlet_id=b.sales_outlet_id) as t1
on a.staff_id=t1.manager
```

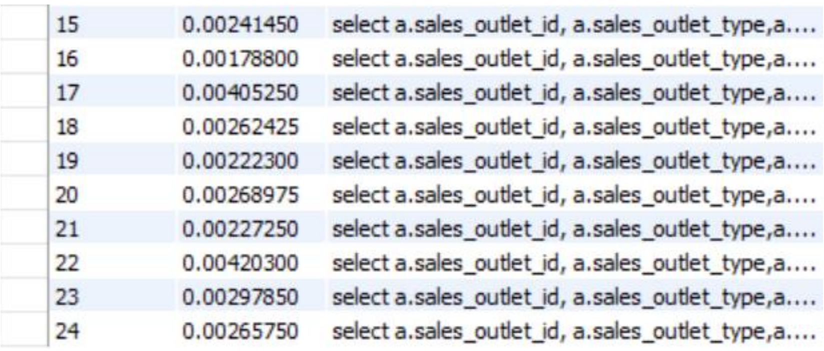
first_name	last_name	sales_outlet_id
Xena	Rahim	3
Ruth	Leslie	4
Reed	Eve	5
Melodie	Mercedes	6
Joelle	Christen	7
Dawn	Anthony	8
Anthony	Kaitlin	9
Adrian	Macon	10

1.5. Execute and time these queries on both databases

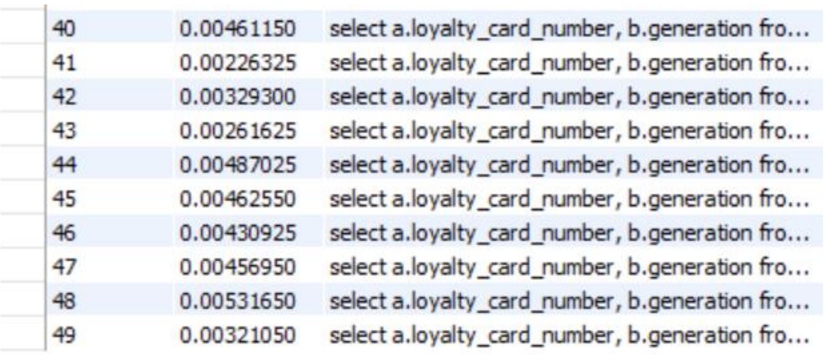
Query timings in Database A:



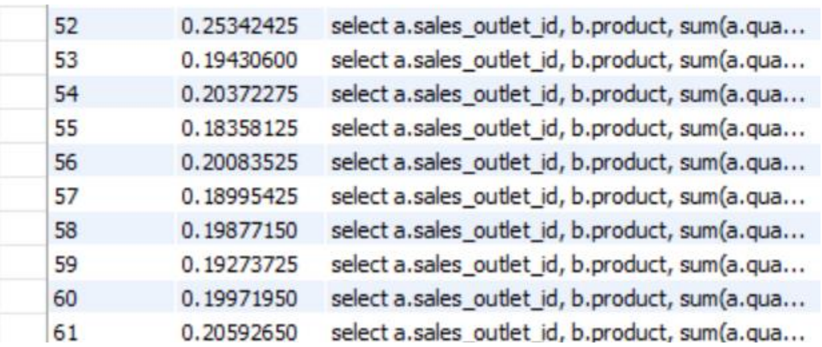
27	0.00174250	select a.product, a.product_category, sum(b.w...
28	0.00326050	select a.product, a.product_category, sum(b.w...
29	0.00251700	select a.product, a.product_category, sum(b.w...
30	0.00286700	select a.product, a.product_category, sum(b.w...
31	0.00312600	select a.product, a.product_category, sum(b.w...
32	0.00309050	select a.product, a.product_category, sum(b.w...
33	0.00395650	select a.product, a.product_category, sum(b.w...
34	0.00396300	select a.product, a.product_category, sum(b.w...
35	0.00313300	select a.product, a.product_category, sum(b.w...
36	0.00237125	select a.product, a.product_category, sum(b.w...



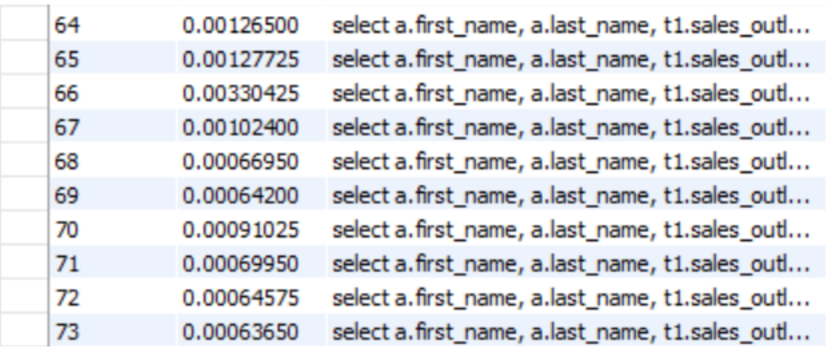
15	0.00241450	select a.sales_outlet_id, a.sales_outlet_type,a....
16	0.00178800	select a.sales_outlet_id, a.sales_outlet_type,a....
17	0.00405250	select a.sales_outlet_id, a.sales_outlet_type,a....
18	0.00262425	select a.sales_outlet_id, a.sales_outlet_type,a....
19	0.00222300	select a.sales_outlet_id, a.sales_outlet_type,a....
20	0.00268975	select a.sales_outlet_id, a.sales_outlet_type,a....
21	0.00227250	select a.sales_outlet_id, a.sales_outlet_type,a....
22	0.00420300	select a.sales_outlet_id, a.sales_outlet_type,a....
23	0.00297850	select a.sales_outlet_id, a.sales_outlet_type,a....
24	0.00265750	select a.sales_outlet_id, a.sales_outlet_type,a....



40	0.00461150	select a.loyalty_card_number, b.generation fro...
41	0.00226325	select a.loyalty_card_number, b.generation fro...
42	0.00329300	select a.loyalty_card_number, b.generation fro...
43	0.00261625	select a.loyalty_card_number, b.generation fro...
44	0.00487025	select a.loyalty_card_number, b.generation fro...
45	0.00462550	select a.loyalty_card_number, b.generation fro...
46	0.00430925	select a.loyalty_card_number, b.generation fro...
47	0.00456950	select a.loyalty_card_number, b.generation fro...
48	0.00531650	select a.loyalty_card_number, b.generation fro...
49	0.00321050	select a.loyalty_card_number, b.generation fro...

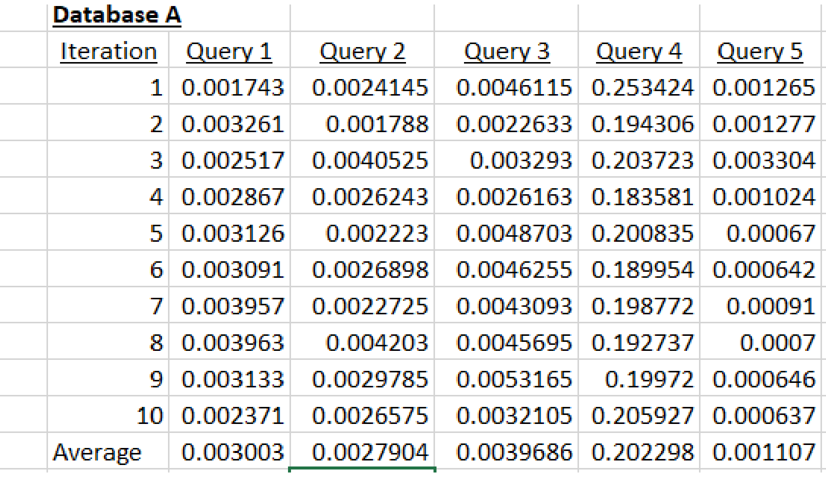


52	0.25342425	select a.sales_outlet_id, b.product, sum(a.qua...
53	0.19430600	select a.sales_outlet_id, b.product, sum(a.qua...
54	0.20372275	select a.sales_outlet_id, b.product, sum(a.qua...
55	0.18358125	select a.sales_outlet_id, b.product, sum(a.qua...
56	0.20083525	select a.sales_outlet_id, b.product, sum(a.qua...
57	0.18995425	select a.sales_outlet_id, b.product, sum(a.qua...
58	0.19877150	select a.sales_outlet_id, b.product, sum(a.qua...
59	0.19273725	select a.sales_outlet_id, b.product, sum(a.qua...
60	0.19971950	select a.sales_outlet_id, b.product, sum(a.qua...
61	0.20592650	select a.sales_outlet_id, b.product, sum(a.qua...



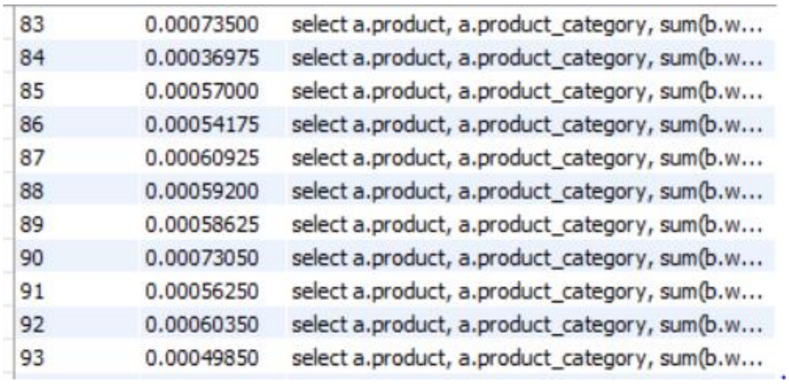
64	0.00126500	select a.first_name, a.last_name, t1.sales_outl...
65	0.00127725	select a.first_name, a.last_name, t1.sales_outl...
66	0.00330425	select a.first_name, a.last_name, t1.sales_outl...
67	0.00102400	select a.first_name, a.last_name, t1.sales_outl...
68	0.00066950	select a.first_name, a.last_name, t1.sales_outl...
69	0.00064200	select a.first_name, a.last_name, t1.sales_outl...
70	0.00091025	select a.first_name, a.last_name, t1.sales_outl...
71	0.00069950	select a.first_name, a.last_name, t1.sales_outl...
72	0.00064575	select a.first_name, a.last_name, t1.sales_outl...
73	0.00063650	select a.first_name, a.last_name, t1.sales_outl...

Summary of query timing on Database A

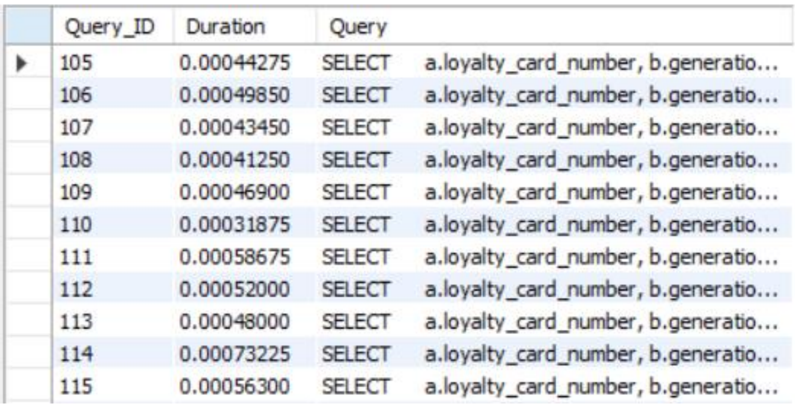


Database A					
Iteration	Query 1	Query 2	Query 3	Query 4	Query 5
1	0.001743	0.0024145	0.0046115	0.253424	0.001265
2	0.003261	0.001788	0.0022633	0.194306	0.001277
3	0.002517	0.0040525	0.003293	0.203723	0.003304
4	0.002867	0.0026243	0.0026163	0.183581	0.001024
5	0.003126	0.002223	0.0048703	0.200835	0.00067
6	0.003091	0.0026898	0.0046255	0.189954	0.000642
7	0.003957	0.0022725	0.0043093	0.198772	0.00091
8	0.003963	0.004203	0.0045695	0.192737	0.0007
9	0.003133	0.0029785	0.0053165	0.19972	0.000646
10	0.002371	0.0026575	0.0032105	0.205927	0.000637
Average	0.003003	0.0027904	0.0039686	0.202298	0.001107

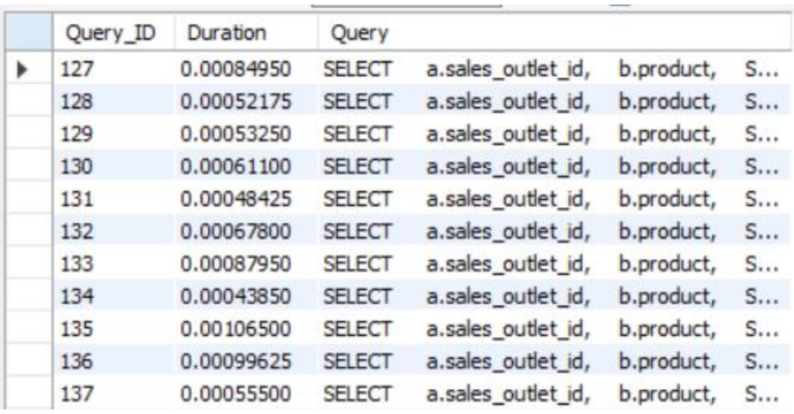
Query timings in Database B:



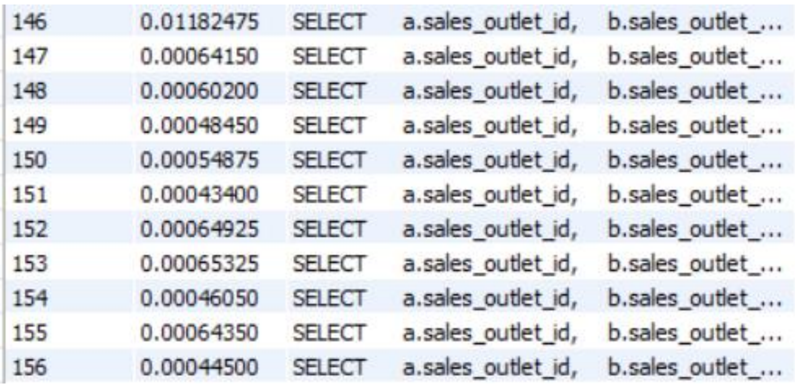
83	0.00073500	select a.product, a.product_category, sum(b.w...
84	0.00036975	select a.product, a.product_category, sum(b.w...
85	0.00057000	select a.product, a.product_category, sum(b.w...
86	0.00054175	select a.product, a.product_category, sum(b.w...
87	0.00060925	select a.product, a.product_category, sum(b.w...
88	0.00059200	select a.product, a.product_category, sum(b.w...
89	0.00058625	select a.product, a.product_category, sum(b.w...
90	0.00073050	select a.product, a.product_category, sum(b.w...
91	0.00056250	select a.product, a.product_category, sum(b.w...
92	0.00060350	select a.product, a.product_category, sum(b.w...
93	0.00049850	select a.product, a.product_category, sum(b.w...



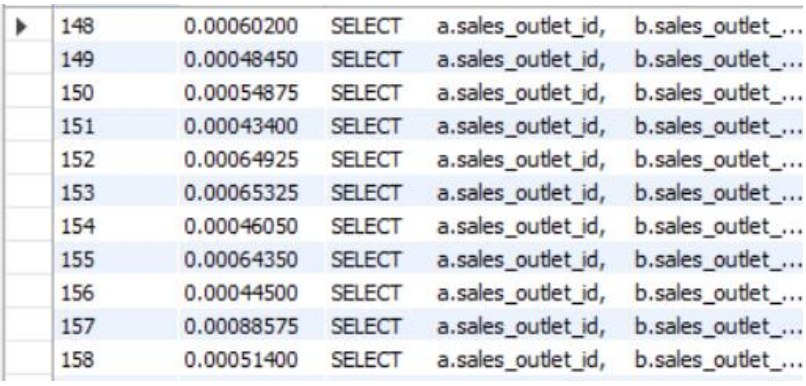
Query_ID	Duration	Query
105	0.00044275	SELECT a.loyalty_card_number, b.generatio...
106	0.00049850	SELECT a.loyalty_card_number, b.generatio...
107	0.00043450	SELECT a.loyalty_card_number, b.generatio...
108	0.00041250	SELECT a.loyalty_card_number, b.generatio...
109	0.00046900	SELECT a.loyalty_card_number, b.generatio...
110	0.00031875	SELECT a.loyalty_card_number, b.generatio...
111	0.00058675	SELECT a.loyalty_card_number, b.generatio...
112	0.00052000	SELECT a.loyalty_card_number, b.generatio...
113	0.00048000	SELECT a.loyalty_card_number, b.generatio...
114	0.00073225	SELECT a.loyalty_card_number, b.generatio...
115	0.00056300	SELECT a.loyalty_card_number, b.generatio...



Query_ID	Duration	Query
127	0.00084950	SELECT a.sales_outlet_id, b.product, S...
128	0.00052175	SELECT a.sales_outlet_id, b.product, S...
129	0.00053250	SELECT a.sales_outlet_id, b.product, S...
130	0.00061100	SELECT a.sales_outlet_id, b.product, S...
131	0.00048425	SELECT a.sales_outlet_id, b.product, S...
132	0.00067800	SELECT a.sales_outlet_id, b.product, S...
133	0.00087950	SELECT a.sales_outlet_id, b.product, S...
134	0.00043850	SELECT a.sales_outlet_id, b.product, S...
135	0.00106500	SELECT a.sales_outlet_id, b.product, S...
136	0.00099625	SELECT a.sales_outlet_id, b.product, S...
137	0.00055500	SELECT a.sales_outlet_id, b.product, S...

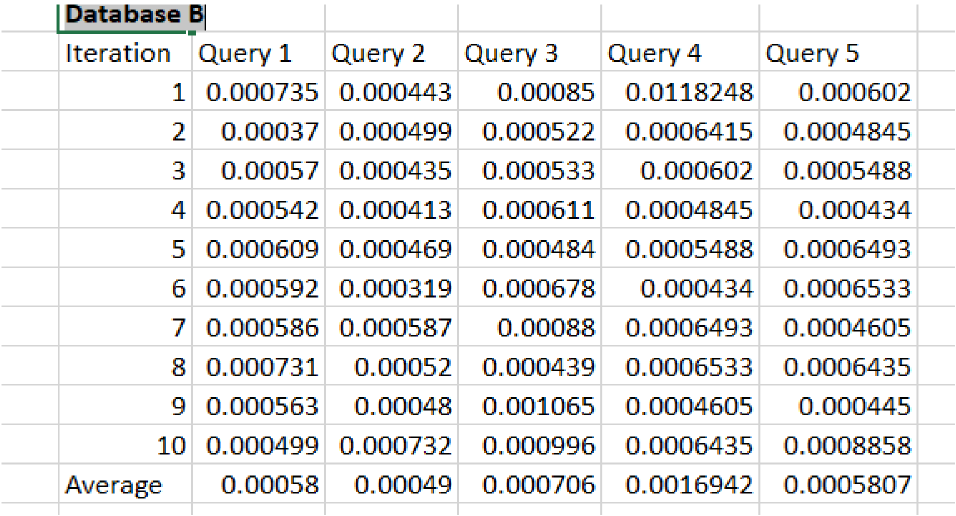


146	0.01182475	SELECT a.sales_outlet_id, b.sales_outlet_...
147	0.00064150	SELECT a.sales_outlet_id, b.sales_outlet_...
148	0.00060200	SELECT a.sales_outlet_id, b.sales_outlet_...
149	0.00048450	SELECT a.sales_outlet_id, b.sales_outlet_...
150	0.00054875	SELECT a.sales_outlet_id, b.sales_outlet_...
151	0.00043400	SELECT a.sales_outlet_id, b.sales_outlet_...
152	0.00064925	SELECT a.sales_outlet_id, b.sales_outlet_...
153	0.00065325	SELECT a.sales_outlet_id, b.sales_outlet_...
154	0.00046050	SELECT a.sales_outlet_id, b.sales_outlet_...
155	0.00064350	SELECT a.sales_outlet_id, b.sales_outlet_...
156	0.00044500	SELECT a.sales_outlet_id, b.sales_outlet_...



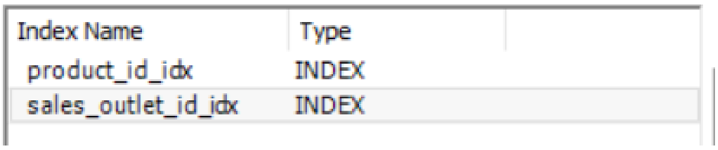
148	0.00060200	SELECT a.sales_outlet_id, b.sales_outlet_...
149	0.00048450	SELECT a.sales_outlet_id, b.sales_outlet_...
150	0.00054875	SELECT a.sales_outlet_id, b.sales_outlet_...
151	0.00043400	SELECT a.sales_outlet_id, b.sales_outlet_...
152	0.00064925	SELECT a.sales_outlet_id, b.sales_outlet_...
153	0.00065325	SELECT a.sales_outlet_id, b.sales_outlet_...
154	0.00046050	SELECT a.sales_outlet_id, b.sales_outlet_...
155	0.00064350	SELECT a.sales_outlet_id, b.sales_outlet_...
156	0.00044500	SELECT a.sales_outlet_id, b.sales_outlet_...
157	0.00088575	SELECT a.sales_outlet_id, b.sales_outlet_...
158	0.00051400	SELECT a.sales_outlet_id, b.sales_outlet_...

Summary of query timing on Database

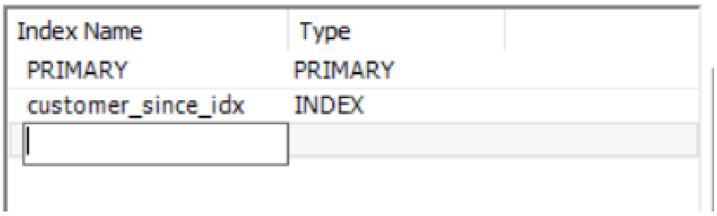


Database B					
Iteration	Query 1	Query 2	Query 3	Query 4	Query 5
1	0.000735	0.000443	0.00085	0.0118248	0.000602
2	0.00037	0.000499	0.000522	0.0006415	0.0004845
3	0.00057	0.000435	0.000533	0.000602	0.0005488
4	0.000542	0.000413	0.000611	0.0004845	0.000434
5	0.000609	0.000469	0.000484	0.0005488	0.0006493
6	0.000592	0.000319	0.000678	0.000434	0.0006533
7	0.000586	0.000587	0.00088	0.0006493	0.0004605
8	0.000731	0.00052	0.000439	0.0006533	0.0006435
9	0.000563	0.00048	0.001065	0.0004605	0.000445
10	0.000499	0.000732	0.000996	0.0006435	0.0008858
Average	0.00058	0.00049	0.000706	0.0016942	0.0005807

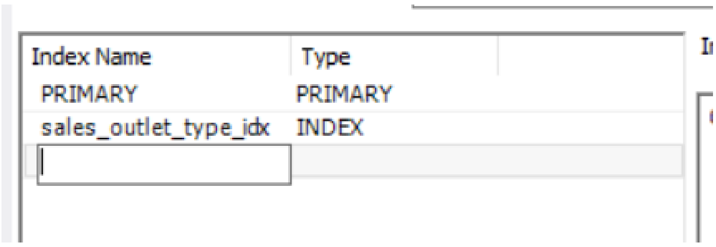
1.6. Add indexes on columns from Database A



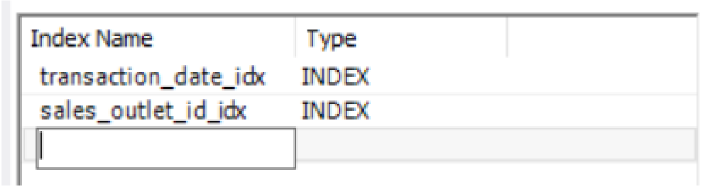
Index Name	Type
product_id_idx	INDEX
sales_outlet_id_idx	INDEX



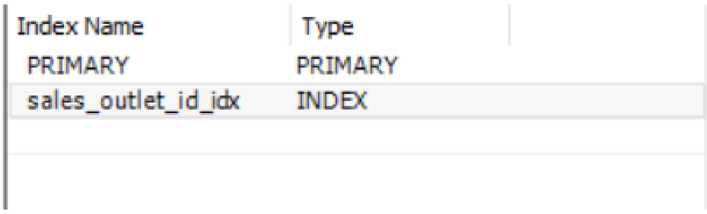
Index Name	Type
PRIMARY	PRIMARY
customer_since_idx	INDEX



Index Name	Type
PRIMARY	PRIMARY
sales_outlet_type_idx	INDEX

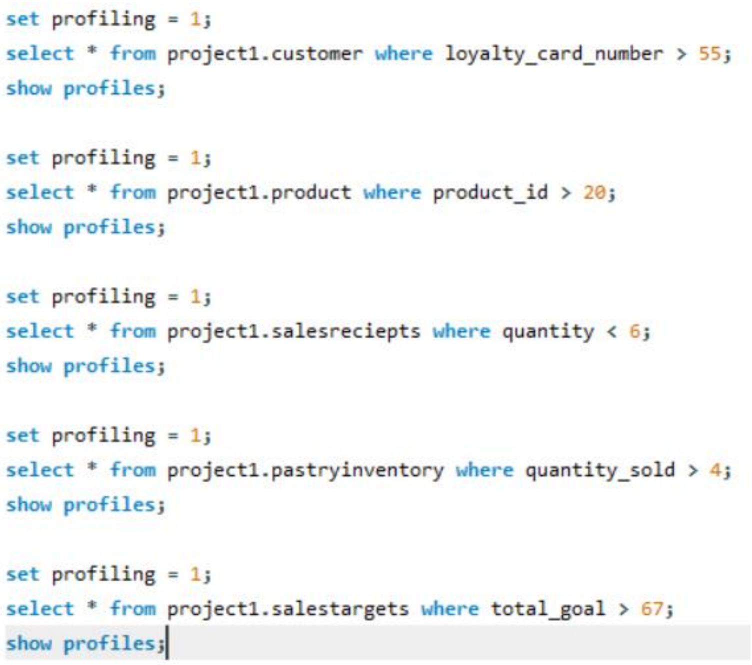


Index Name	Type
transaction_date_idx	INDEX
sales_outlet_id_idx	INDEX



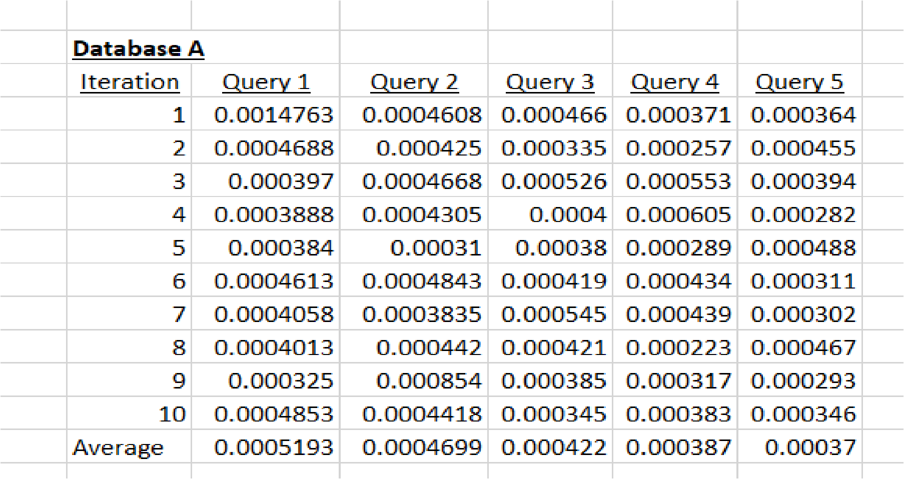
Index Name	Type
PRIMARY	PRIMARY
sales_outlet_id_idx	INDEX

1.7. Write a query for each column

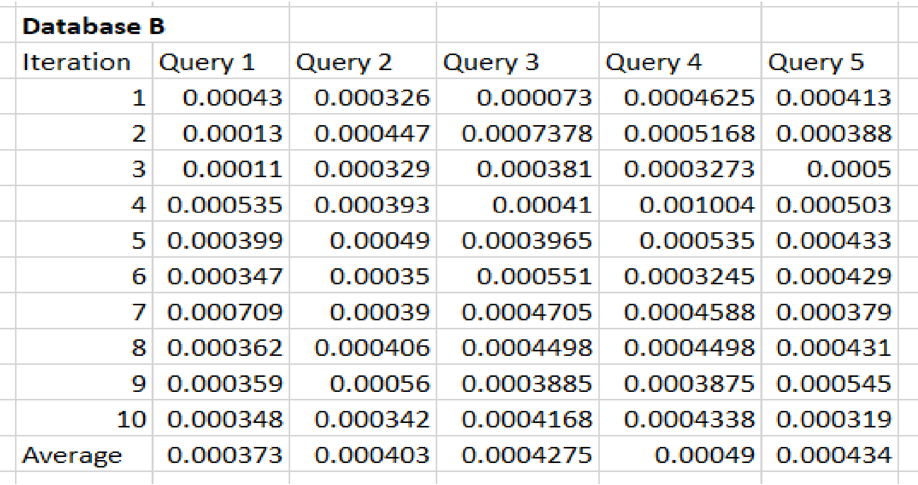


```
set profiling = 1;
select * from project1.customer where loyalty_card_number > 55;
show profiles;

set profiling = 1;
select * from project1.product where product_id > 20;
show profiles;

set profiling = 1;
select * from project1.salesreciepts where quantity < 6;
show profiles;

set profiling = 1;
select * from project1.pastryinventory where quantity_sold > 4;
show profiles;

set profiling = 1;
select * from project1.salestargets where total_goal > 67;
show profiles;
```

1.8. Execute and time these queries on both databases



Database A					
Iteration	Query 1	Query 2	Query 3	Query 4	Query 5
1	0.0014763	0.0004608	0.000466	0.000371	0.000364
2	0.0004688	0.000425	0.000335	0.000257	0.000455
3	0.000397	0.0004668	0.000526	0.000553	0.000394
4	0.0003888	0.0004305	0.0004	0.000605	0.000282
5	0.000384	0.00031	0.00038	0.000289	0.000488
6	0.0004613	0.0004843	0.000419	0.000434	0.000311
7	0.0004058	0.0003835	0.000545	0.000439	0.000302
8	0.0004013	0.000442	0.000421	0.000223	0.000467
9	0.000325	0.000854	0.000385	0.000317	0.000293
10	0.0004853	0.0004418	0.000345	0.000383	0.000346
Average	0.0005193	0.0004699	0.000422	0.000387	0.00037



Database B					
Iteration	Query 1	Query 2	Query 3	Query 4	Query 5
1	0.00043	0.000326	0.000073	0.0004625	0.000413
2	0.00013	0.000447	0.0007378	0.0005168	0.000388
3	0.00011	0.000329	0.000381	0.0003273	0.0005
4	0.000535	0.000393	0.00041	0.001004	0.000503
5	0.000399	0.00049	0.0003965	0.000535	0.000433
6	0.000347	0.00035	0.000551	0.0003245	0.000429
7	0.000709	0.00039	0.0004705	0.0004588	0.000379
8	0.000362	0.000406	0.0004498	0.0004498	0.000431
9	0.000359	0.00056	0.0003885	0.0003875	0.000545
10	0.000348	0.000342	0.0004168	0.0004338	0.000319
Average	0.000373	0.000403	0.0004275	0.00049	0.000434

1.9. Execute and time these queries on both databases

Based off the runtime for both databases, we noticed that the query time for database b was overall significantly faster than database a. Generally, an indexed database should run faster as the data is stored in B TREE structure. We can in the first instance and second instance, our queries run significantly faster in database B than database A. That may be due to the queries not utilizing the indexes set in database a, especially since we've barely run commands to retrieve data so SQL may have not optimized indexes for the database. In the second instance, with adding more indexes to database A we can see that the queries are running at a significantly decreased duration.

Part 2: MongoDB and MQL

2.1 Explore your dataset and familiarize yourself with the dataset and its content

We chose the Bikes dataset for our analysis. We downloaded the file in JSON format, and analysed the different attributes the dataset included and the type of data provided.

2.2 Why it is better to use non-relational databases such as MongoDB to work with such a dataset?

This dataset includes too many attributes for each bike model which makes it computationally difficult to represent as a relational database. To be able to represent it in tabular form, multiple joins will have to be run which is cumbersome and inefficient.

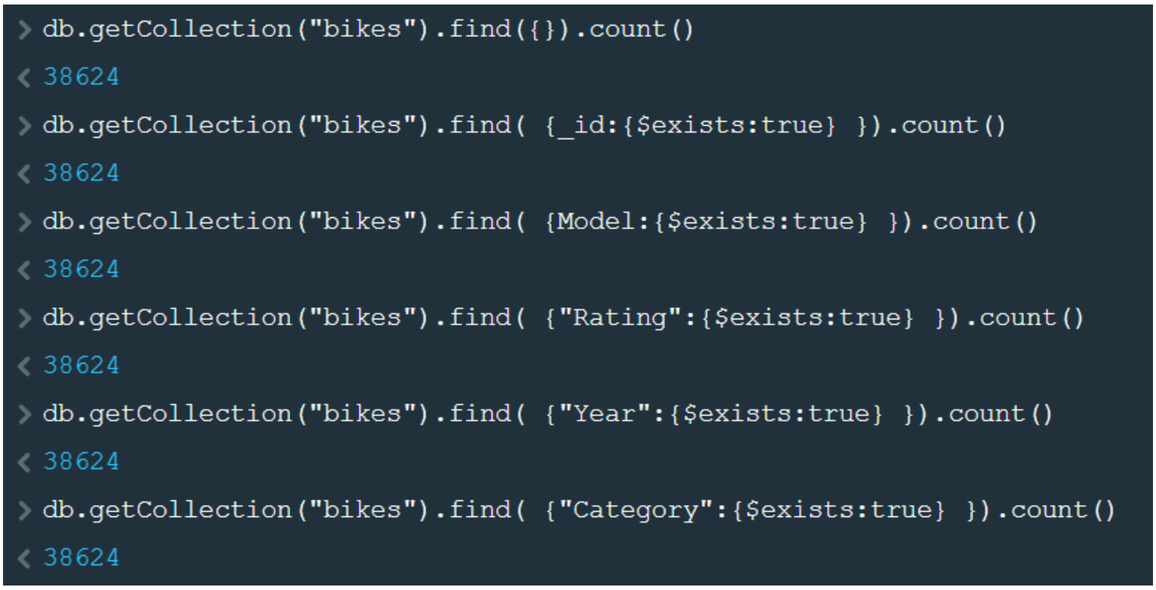
Non-relational databases like MongoDB enables the storage of large amounts of data in a non-tabular form and is computationally more efficient. MongoDB facilitates the storage of data in unstructured, semi-structured, or structured forms and is developer friendly. It is also easier to update schemas and fields.

2.3 We imported the data based on the instructions provided

2.4 List of some of the attributes (field/properties) of our database which are common among all documents:

- _id
- Model
- Rating
- Year
- Category

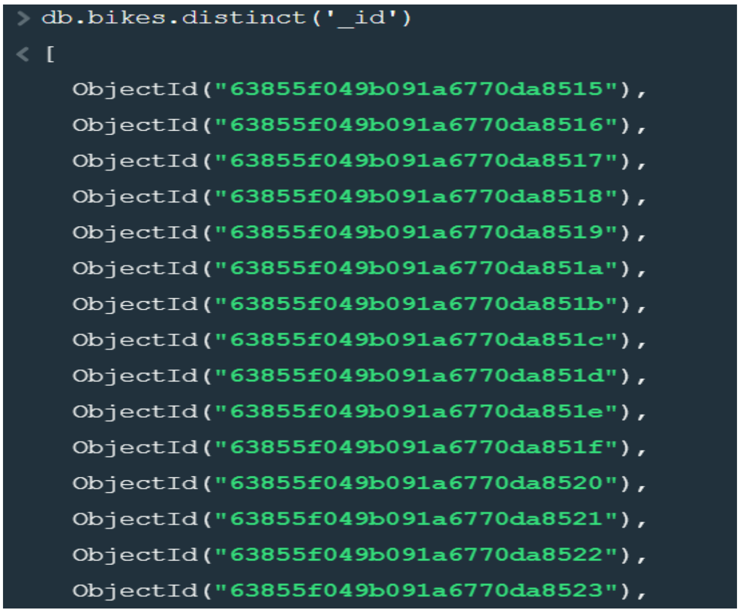
The above mentioned five attributes were common among all the 38,624 documents in the datatset.



```
> db.getCollection("bikes").find({}).count()
< 38624
> db.getCollection("bikes").find( {_id:{$exists:true} }).count()
< 38624
> db.getCollection("bikes").find( {Model:{$exists:true} }).count()
< 38624
> db.getCollection("bikes").find( {"Rating":{$exists:true} }).count()
< 38624
> db.getCollection("bikes").find( {"Year":{$exists:true} }).count()
< 38624
> db.getCollection("bikes").find( {"Category":{$exists:true} }).count()
< 38624
```

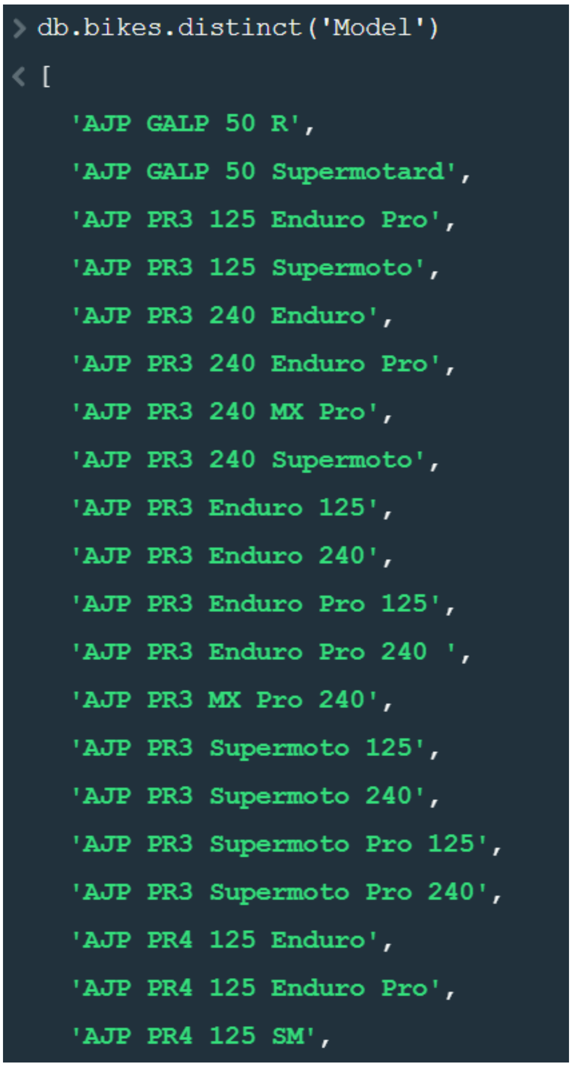
2.4.1 For these fields, provide some of the values they contain in the database

- _id



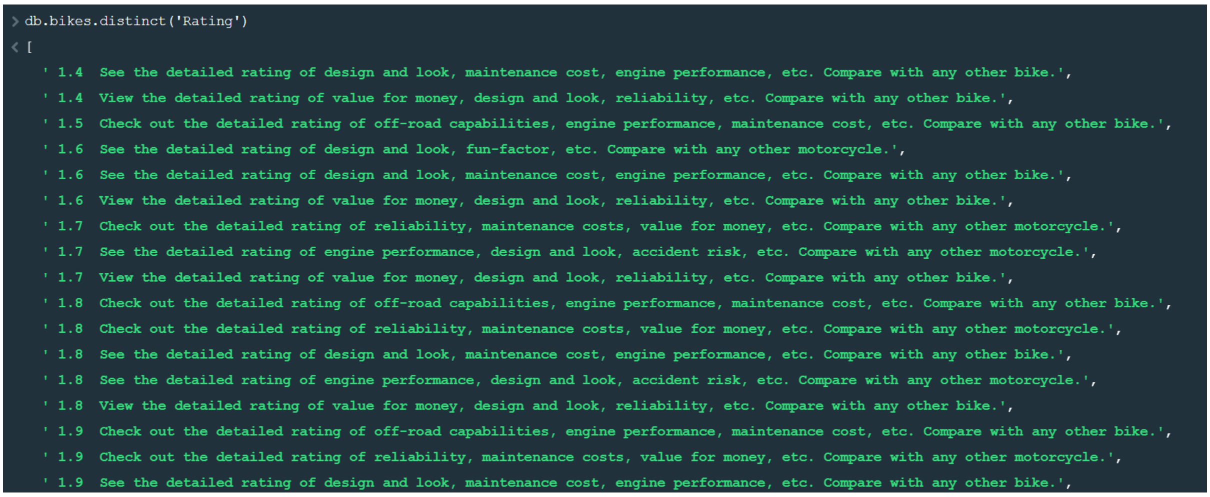
```
> db.bikes.distinct('_id')
< [
    ObjectId("63855f049b091a6770da8515"),
    ObjectId("63855f049b091a6770da8516"),
    ObjectId("63855f049b091a6770da8517"),
    ObjectId("63855f049b091a6770da8518"),
    ObjectId("63855f049b091a6770da8519"),
    ObjectId("63855f049b091a6770da851a"),
    ObjectId("63855f049b091a6770da851b"),
    ObjectId("63855f049b091a6770da851c"),
    ObjectId("63855f049b091a6770da851d"),
    ObjectId("63855f049b091a6770da851e"),
    ObjectId("63855f049b091a6770da851f"),
    ObjectId("63855f049b091a6770da8520"),
    ObjectId("63855f049b091a6770da8521"),
    ObjectId("63855f049b091a6770da8522"),
    ObjectId("63855f049b091a6770da8523"),
```

- Model



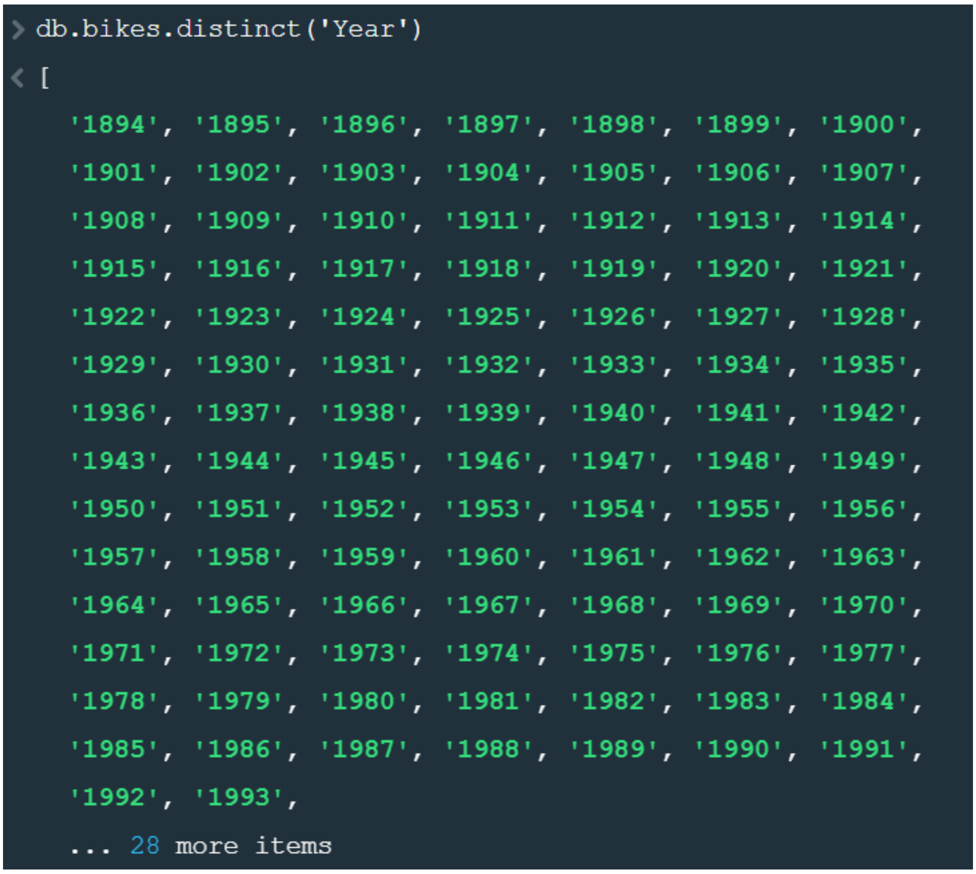
```
> db.bikes.distinct('Model')
< [
    'AJP GALP 50 R',
    'AJP GALP 50 Supermotard',
    'AJP PR3 125 Enduro Pro',
    'AJP PR3 125 Supermoto',
    'AJP PR3 240 Enduro',
    'AJP PR3 240 Enduro Pro',
    'AJP PR3 240 MX Pro',
    'AJP PR3 240 Supermoto',
    'AJP PR3 Enduro 125',
    'AJP PR3 Enduro 240',
    'AJP PR3 Enduro Pro 125',
    'AJP PR3 Enduro Pro 240 ',
    'AJP PR3 MX Pro 240',
    'AJP PR3 Supermoto 125',
    'AJP PR3 Supermoto 240',
    'AJP PR3 Supermoto Pro 125',
    'AJP PR3 Supermoto Pro 240',
    'AJP PR4 125 Enduro',
    'AJP PR4 125 Enduro Pro',
    'AJP PR4 125 SM',
```

- Rating



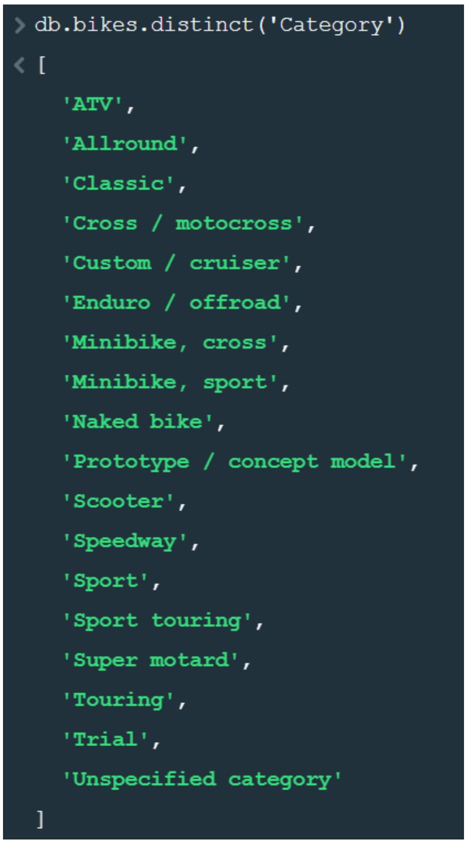
```
> db.bikes.distinct('Rating')
< [
    ' 1.4  See the detailed rating of design and look, maintenance cost, engine performance, etc. Compare with any other bike.',
    ' 1.4  View the detailed rating of value for money, design and look, reliability, etc. Compare with any other bike.',
    ' 1.5  Check out the detailed rating of off-road capabilities, engine performance, maintenance cost, etc. Compare with any other bike.',
    ' 1.6  See the detailed rating of design and look, fun-factor, etc. Compare with any other motorcycle.',
    ' 1.6  See the detailed rating of design and look, maintenance cost, engine performance, etc. Compare with any other bike.',
    ' 1.6  View the detailed rating of value for money, design and look, reliability, etc. Compare with any other bike.',
    ' 1.7  Check out the detailed rating of reliability, maintenance costs, value for money, etc. Compare with any other motorcycle.',
    ' 1.7  See the detailed rating of engine performance, design and look, accident risk, etc. Compare with any other motorcycle.',
    ' 1.7  View the detailed rating of value for money, design and look, reliability, etc. Compare with any other bike.',
    ' 1.8  Check out the detailed rating of off-road capabilities, engine performance, maintenance cost, etc. Compare with any other bike.',
    ' 1.8  Check out the detailed rating of reliability, maintenance costs, value for money, etc. Compare with any other motorcycle.',
    ' 1.8  See the detailed rating of design and look, maintenance cost, engine performance, etc. Compare with any other bike.',
    ' 1.8  See the detailed rating of engine performance, design and look, accident risk, etc. Compare with any other motorcycle.',
    ' 1.8  View the detailed rating of value for money, design and look, reliability, etc. Compare with any other bike.',
    ' 1.9  Check out the detailed rating of off-road capabilities, engine performance, maintenance cost, etc. Compare with any other bike.',
    ' 1.9  Check out the detailed rating of reliability, maintenance costs, value for money, etc. Compare with any other motorcycle.',
    ' 1.9  See the detailed rating of design and look, maintenance cost, engine performance, etc. Compare with any other bike.',
```

- Year



```
> db.bikes.distinct('Year')
< [
    '1894', '1895', '1896', '1897', '1898', '1899', '1900',
    '1901', '1902', '1903', '1904', '1905', '1906', '1907',
    '1908', '1909', '1910', '1911', '1912', '1913', '1914',
    '1915', '1916', '1917', '1918', '1919', '1920', '1921',
    '1922', '1923', '1924', '1925', '1926', '1927', '1928',
    '1929', '1930', '1931', '1932', '1933', '1934', '1935',
    '1936', '1937', '1938', '1939', '1940', '1941', '1942',
    '1943', '1944', '1945', '1946', '1947', '1948', '1949',
    '1950', '1951', '1952', '1953', '1954', '1955', '1956',
    '1957', '1958', '1959', '1960', '1961', '1962', '1963',
    '1964', '1965', '1966', '1967', '1968', '1969', '1970',
    '1971', '1972', '1973', '1974', '1975', '1976', '1977',
    '1978', '1979', '1980', '1981', '1982', '1983', '1984',
    '1985', '1986', '1987', '1988', '1989', '1990', '1991',
    '1992', '1993',
    ... 28 more items
```

- Category

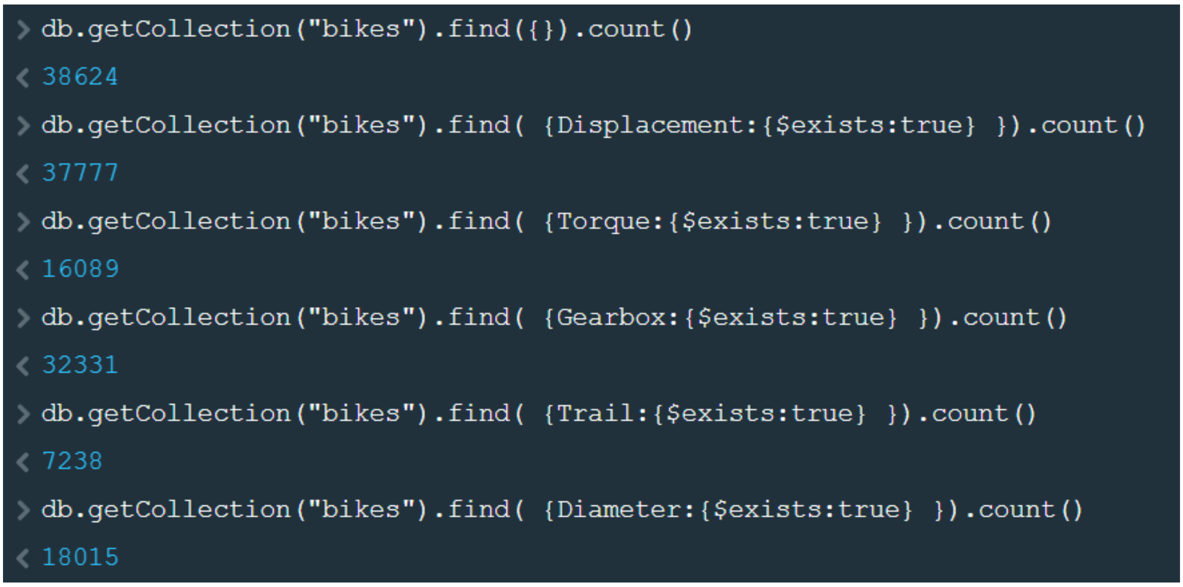


```
> db.bikes.distinct('Category')
< [
    'ATV',
    'Allround',
    'Classic',
    'Cross / motocross',
    'Custom / cruiser',
    'Enduro / offroad',
    'Minibike, cross',
    'Minibike, sport',
    'Naked bike',
    'Prototype / concept model',
    'Scooter',
    'Speedway',
    'Sport',
    'Sport touring',
    'Super motard',
    'Touring',
    'Trial',
    'Unspecified category'
  ]
```

2.5 List some of the attributes (field/properties) of your database which are not common among all the documents

- Displacement
- Torque
- Gearbox
- Trail
- Diameter

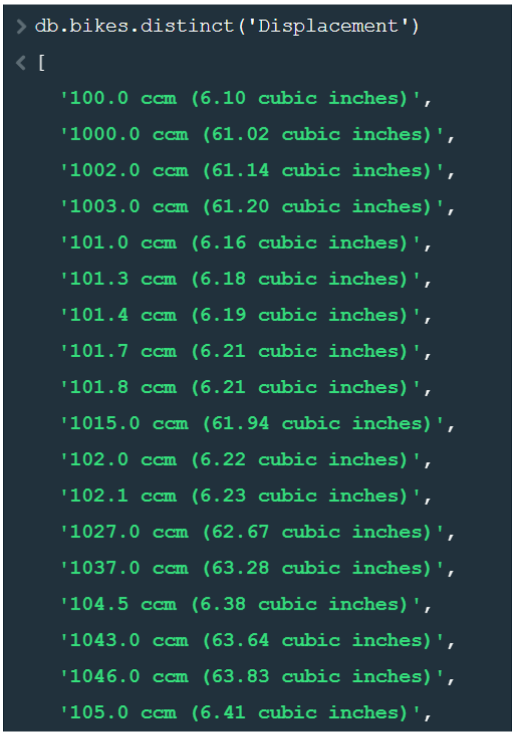
The total number of documents are 38,624 and the above mentioned attributes have a count that is less than 38,624, hence, these attributes are not common among all the documents.



```
> db.getCollection("bikes").find({}).count()
< 38624
> db.getCollection("bikes").find( {Displacement:{$exists:true} }).count()
< 37777
> db.getCollection("bikes").find( {Torque:{$exists:true} }).count()
< 16089
> db.getCollection("bikes").find( {Gearbox:{$exists:true} }).count()
< 32331
> db.getCollection("bikes").find( {Trail:{$exists:true} }).count()
< 7238
> db.getCollection("bikes").find( {Diameter:{$exists:true} }).count()
< 18015
```

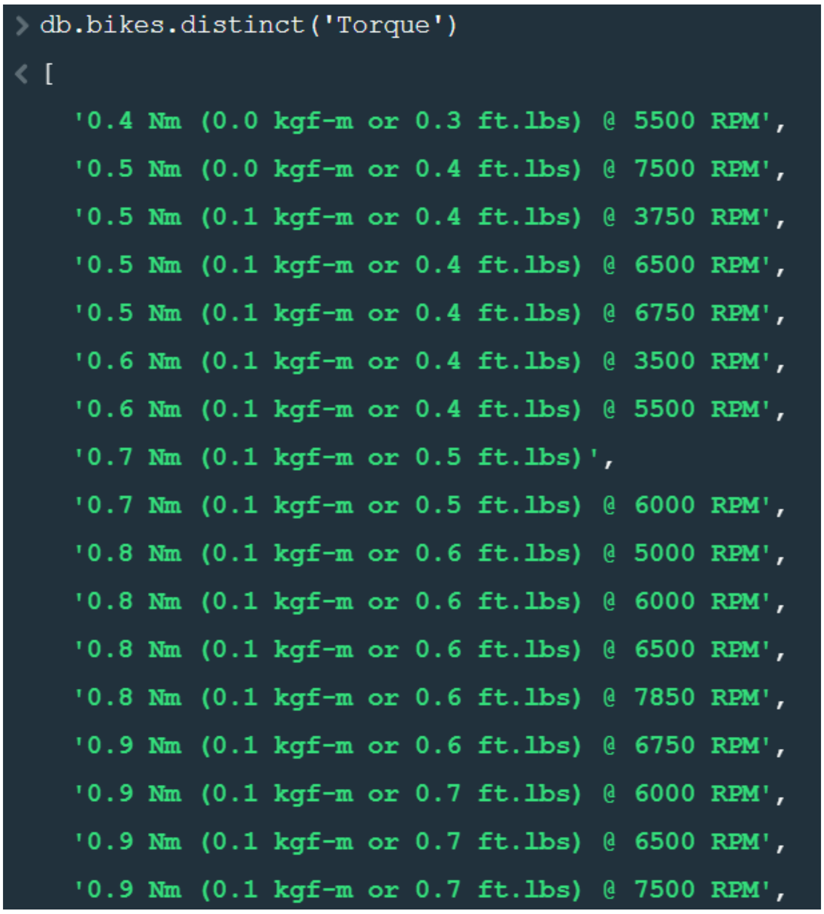
2.5.1 For these fields, provide some of the values they contain in the database

- Displacement



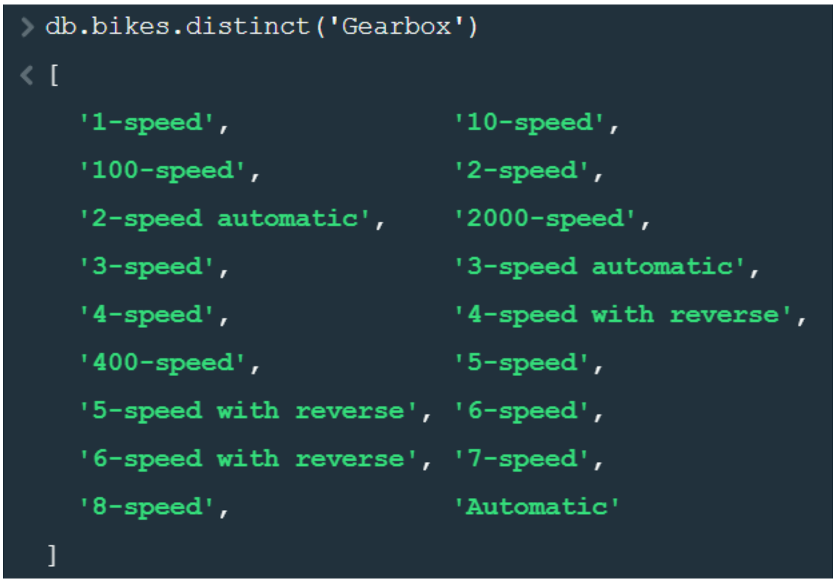
```
> db.bikes.distinct('Displacement')
< [
    '100.0 ccm (6.10 cubic inches)',
    '1000.0 ccm (61.02 cubic inches)',
    '1002.0 ccm (61.14 cubic inches)',
    '1003.0 ccm (61.20 cubic inches)',
    '101.0 ccm (6.16 cubic inches)',
    '101.3 ccm (6.18 cubic inches)',
    '101.4 ccm (6.19 cubic inches)',
    '101.7 ccm (6.21 cubic inches)',
    '101.8 ccm (6.21 cubic inches)',
    '1015.0 ccm (61.94 cubic inches)',
    '102.0 ccm (6.22 cubic inches)',
    '102.1 ccm (6.23 cubic inches)',
    '1027.0 ccm (62.67 cubic inches)',
    '1037.0 ccm (63.28 cubic inches)',
    '104.5 ccm (6.38 cubic inches)',
    '1043.0 ccm (63.64 cubic inches)',
    '1046.0 ccm (63.83 cubic inches)',
    '105.0 ccm (6.41 cubic inches)',
```

- Torque



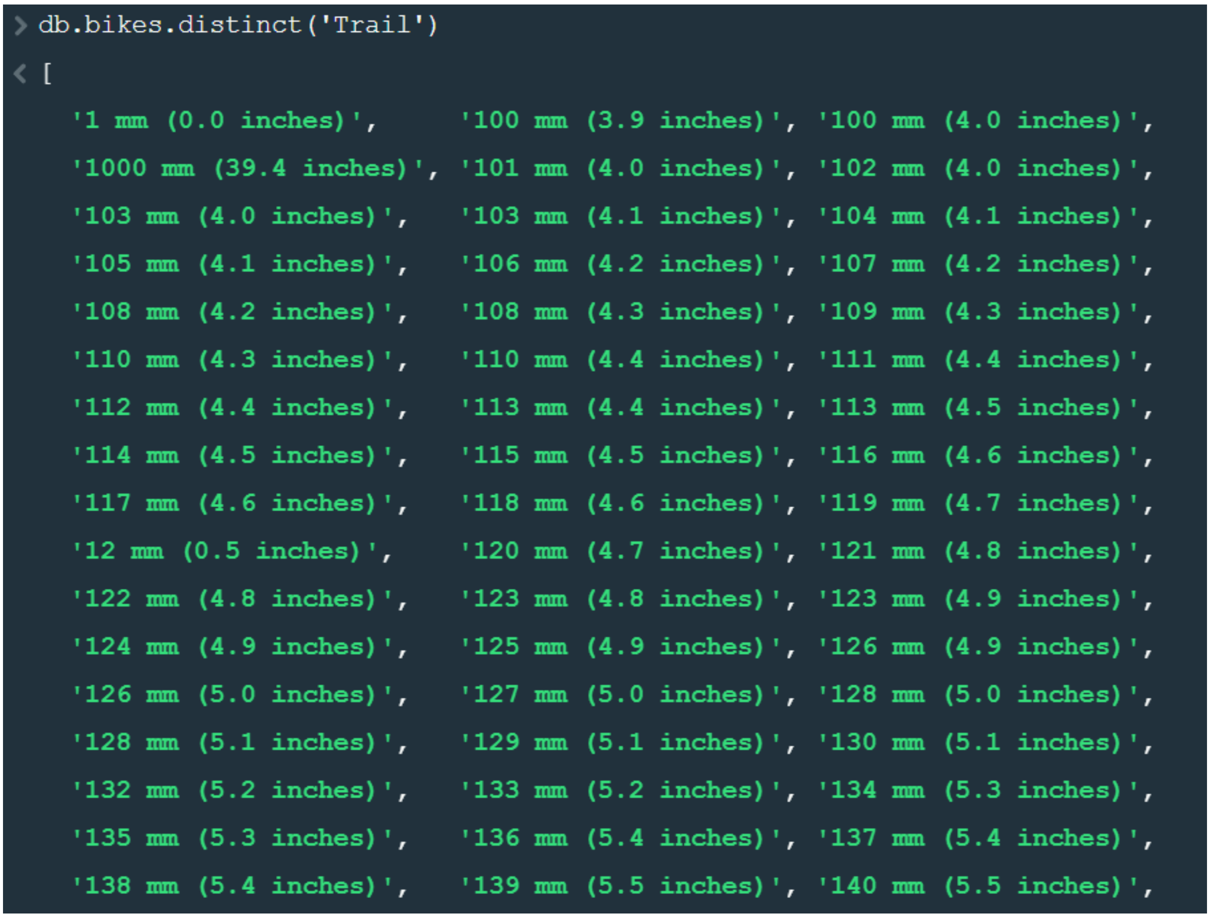
```
> db.bikes.distinct('Torque')
< [
    '0.4 Nm (0.0 kgf-m or 0.3 ft.lbs) @ 5500 RPM',
    '0.5 Nm (0.0 kgf-m or 0.4 ft.lbs) @ 7500 RPM',
    '0.5 Nm (0.1 kgf-m or 0.4 ft.lbs) @ 3750 RPM',
    '0.5 Nm (0.1 kgf-m or 0.4 ft.lbs) @ 6500 RPM',
    '0.5 Nm (0.1 kgf-m or 0.4 ft.lbs) @ 6750 RPM',
    '0.6 Nm (0.1 kgf-m or 0.4 ft.lbs) @ 3500 RPM',
    '0.6 Nm (0.1 kgf-m or 0.4 ft.lbs) @ 5500 RPM',
    '0.7 Nm (0.1 kgf-m or 0.5 ft.lbs)',
    '0.7 Nm (0.1 kgf-m or 0.5 ft.lbs) @ 6000 RPM',
    '0.8 Nm (0.1 kgf-m or 0.6 ft.lbs) @ 5000 RPM',
    '0.8 Nm (0.1 kgf-m or 0.6 ft.lbs) @ 6000 RPM',
    '0.8 Nm (0.1 kgf-m or 0.6 ft.lbs) @ 6500 RPM',
    '0.8 Nm (0.1 kgf-m or 0.6 ft.lbs) @ 7850 RPM',
    '0.9 Nm (0.1 kgf-m or 0.6 ft.lbs) @ 6750 RPM',
    '0.9 Nm (0.1 kgf-m or 0.7 ft.lbs) @ 6000 RPM',
    '0.9 Nm (0.1 kgf-m or 0.7 ft.lbs) @ 6500 RPM',
    '0.9 Nm (0.1 kgf-m or 0.7 ft.lbs) @ 7500 RPM',
```

- Gearbox



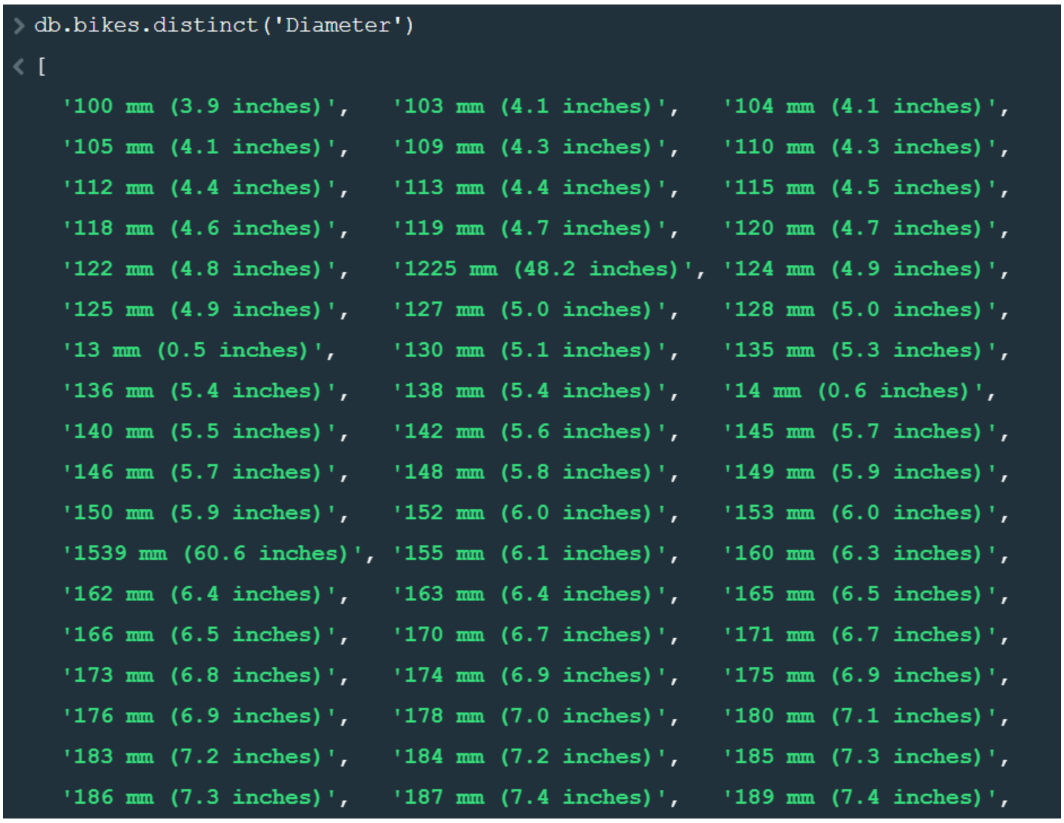
```
> db.bikes.distinct('Gearbox')
< [
    '1-speed',                 '10-speed',
    '100-speed',               '2-speed',
    '2-speed automatic',       '2000-speed',
    '3-speed',                 '3-speed automatic',
    '4-speed',                 '4-speed with reverse',
    '400-speed',               '5-speed',
    '5-speed with reverse',    '6-speed',
    '6-speed with reverse',    '7-speed',
    '8-speed',                 'Automatic'
  ]
```

- Trail



```
> db.bikes.distinct('Trail')
< [
    '1 mm (0.0 inches)',      '100 mm (3.9 inches)', '100 mm (4.0 inches)',
    '1000 mm (39.4 inches)', '101 mm (4.0 inches)', '102 mm (4.0 inches)',
    '103 mm (4.0 inches)',   '103 mm (4.1 inches)', '104 mm (4.1 inches)',
    '105 mm (4.1 inches)',   '106 mm (4.2 inches)', '107 mm (4.2 inches)',
    '108 mm (4.2 inches)',   '108 mm (4.3 inches)', '109 mm (4.3 inches)',
    '110 mm (4.3 inches)',   '110 mm (4.4 inches)', '111 mm (4.4 inches)',
    '112 mm (4.4 inches)',   '113 mm (4.4 inches)', '113 mm (4.5 inches)',
    '114 mm (4.5 inches)',   '115 mm (4.5 inches)', '116 mm (4.6 inches)',
    '117 mm (4.6 inches)',   '118 mm (4.6 inches)', '119 mm (4.7 inches)',
    '12 mm (0.5 inches)',    '120 mm (4.7 inches)', '121 mm (4.8 inches)',
    '122 mm (4.8 inches)',   '123 mm (4.8 inches)', '123 mm (4.9 inches)',
    '124 mm (4.9 inches)',   '125 mm (4.9 inches)', '126 mm (4.9 inches)',
    '126 mm (5.0 inches)',   '127 mm (5.0 inches)', '128 mm (5.0 inches)',
    '128 mm (5.1 inches)',   '129 mm (5.1 inches)', '130 mm (5.1 inches)',
    '132 mm (5.2 inches)',   '133 mm (5.2 inches)', '134 mm (5.3 inches)',
    '135 mm (5.3 inches)',   '136 mm (5.4 inches)', '137 mm (5.4 inches)',
    '138 mm (5.4 inches)',   '139 mm (5.5 inches)', '140 mm (5.5 inches)',
```

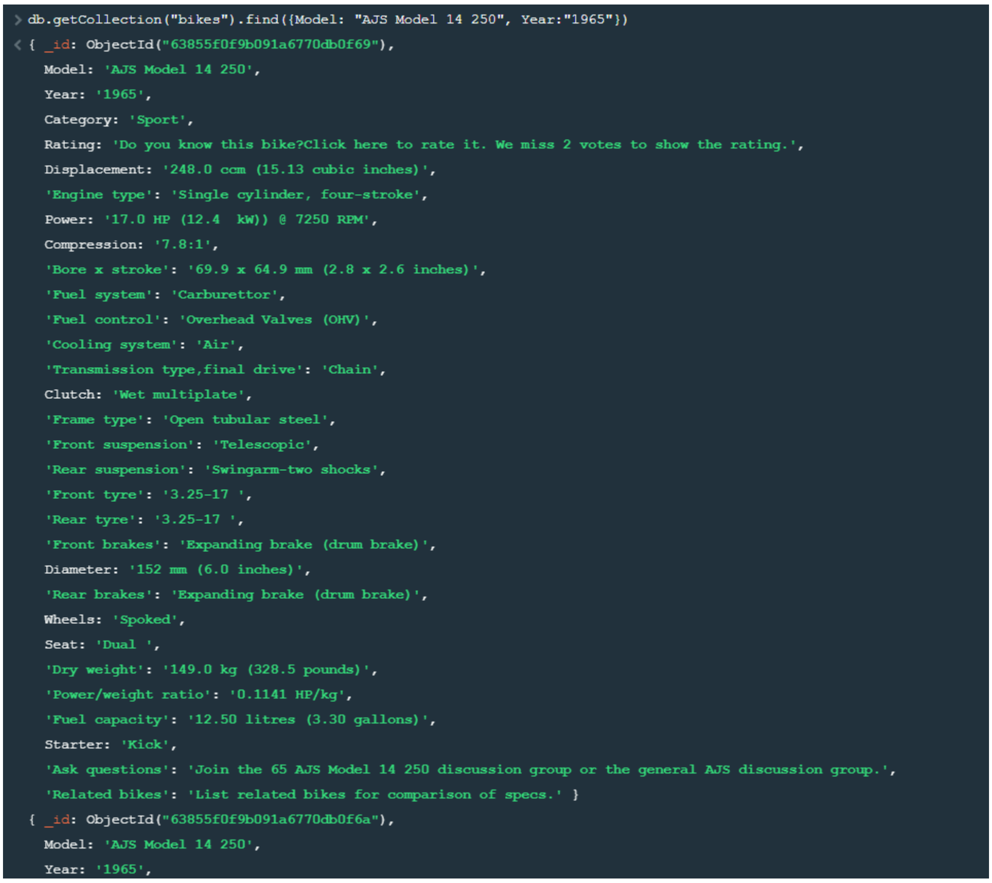
- Diameter



```
> db.bikes.distinct('Diameter')
< [
    '100 mm (3.9 inches)',   '103 mm (4.1 inches)',   '104 mm (4.1 inches)',
    '105 mm (4.1 inches)',   '109 mm (4.3 inches)',   '110 mm (4.3 inches)',
    '112 mm (4.4 inches)',   '113 mm (4.4 inches)',   '115 mm (4.5 inches)',
    '118 mm (4.6 inches)',   '119 mm (4.7 inches)',   '120 mm (4.7 inches)',
    '122 mm (4.8 inches)',   '1225 mm (48.2 inches)', '124 mm (4.9 inches)',
    '125 mm (4.9 inches)',   '127 mm (5.0 inches)',   '128 mm (5.0 inches)',
    '13 mm (0.5 inches)',    '130 mm (5.1 inches)',   '135 mm (5.3 inches)',
    '136 mm (5.4 inches)',   '138 mm (5.4 inches)',   '14 mm (0.6 inches)',
    '140 mm (5.5 inches)',   '142 mm (5.6 inches)',   '145 mm (5.7 inches)',
    '146 mm (5.7 inches)',   '148 mm (5.8 inches)',   '149 mm (5.9 inches)',
    '150 mm (5.9 inches)',   '152 mm (6.0 inches)',   '153 mm (6.0 inches)',
    '1539 mm (60.6 inches)', '155 mm (6.1 inches)',   '160 mm (6.3 inches)',
    '162 mm (6.4 inches)',   '163 mm (6.4 inches)',   '165 mm (6.5 inches)',
    '166 mm (6.5 inches)',   '170 mm (6.7 inches)',   '171 mm (6.7 inches)',
    '173 mm (6.8 inches)',   '174 mm (6.9 inches)',   '175 mm (6.9 inches)',
    '176 mm (6.9 inches)',   '178 mm (7.0 inches)',   '180 mm (7.1 inches)',
    '183 mm (7.2 inches)',   '184 mm (7.2 inches)',   '185 mm (7.3 inches)',
    '186 mm (7.3 inches)',   '187 mm (7.4 inches)',   '189 mm (7.4 inches)',
```

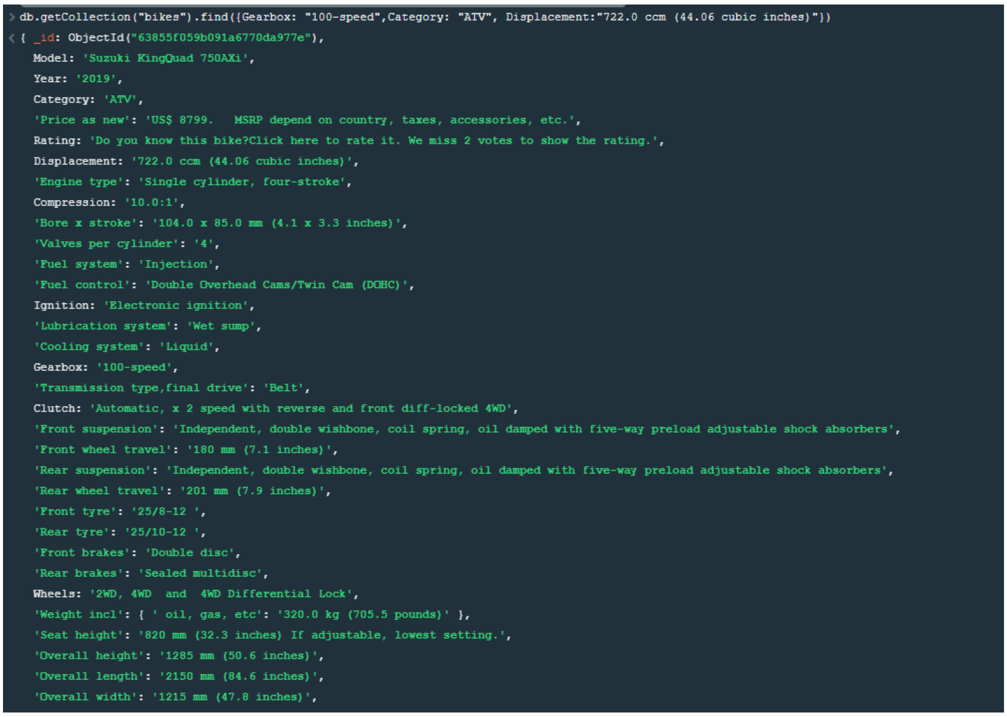
2.6 5 queries using the key-value pairs you found in the previous steps to narrow down the result

1. db.getCollection("bikes").find({Model: "AJS Model 14 250", Year:"1965"})



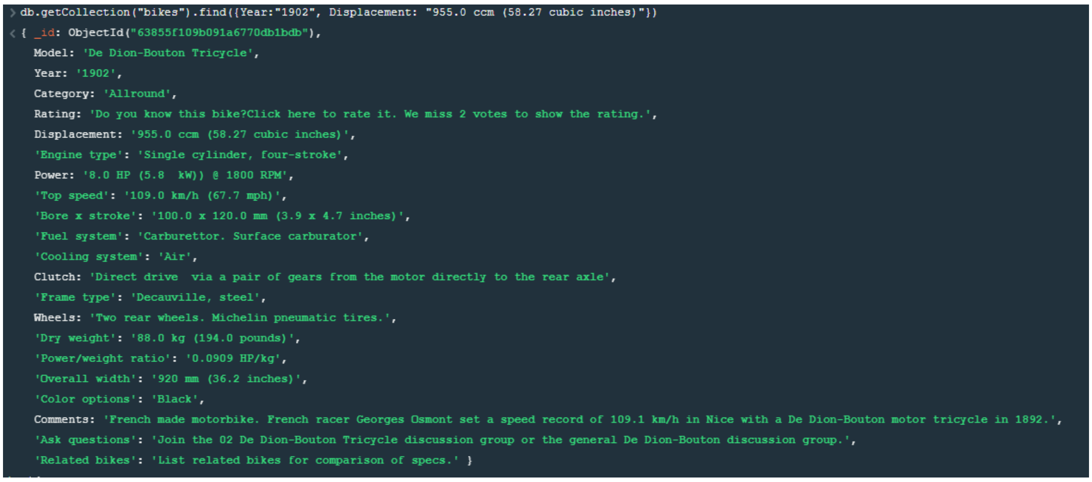
```
> db.getCollection("bikes").find({Model: "AJS Model 14 250", Year:"1965"})
< { _id: ObjectId("63855f0f9b091a6770db0f69"),
    Model: 'AJS Model 14 250',
    Year: '1965',
    Category: 'Sport',
    Rating: 'Do you know this bike?Click here to rate it. We miss 2 votes to show the rating.',
    Displacement: '248.0 ccm (15.13 cubic inches)',
    'Engine type': 'Single cylinder, four-stroke',
    Power: '17.0 HP (12.4  kW)) @ 7250 RPM',
    Compression: '7.8:1',
    'Bore x stroke': '69.9 x 64.9 mm (2.8 x 2.6 inches)',
    'Fuel system': 'Carburettor',
    'Fuel control': 'Overhead Valves (OHV)',
    'Cooling system': 'Air',
    'Transmission type,final drive': 'Chain',
    Clutch: 'Wet multiplate',
    'Frame type': 'Open tubular steel',
    'Front suspension': 'Telescopic',
    'Rear suspension': 'Swingarm-two shocks',
    'Front tyre': '3.25-17 ',
    'Rear tyre': '3.25-17 ',
    'Front brakes': 'Expanding brake (drum brake)',
    Diameter: '152 mm (6.0 inches)',
    'Rear brakes': 'Expanding brake (drum brake)',
    Wheels: 'Spoked',
    Seat: 'Dual ',
    'Dry weight': '149.0 kg (328.5 pounds)',
    'Power/weight ratio': '0.1141 HP/kg',
    'Fuel capacity': '12.50 litres (3.30 gallons)',
    Starter: 'Kick',
    'Ask questions': 'Join the 65 AJS Model 14 250 discussion group or the general AJS discussion group.',
    'Related bikes': 'List related bikes for comparison of specs.' }
  { _id: ObjectId("63855f0f9b091a6770db0f6a"),
    Model: 'AJS Model 14 250',
    Year: '1965',
```

2. db.getCollection("bikes").find({Gearbox: "100-speed",Category: "ATV", Displacement:"722.0 ccm (44.06 cubic inches)"})



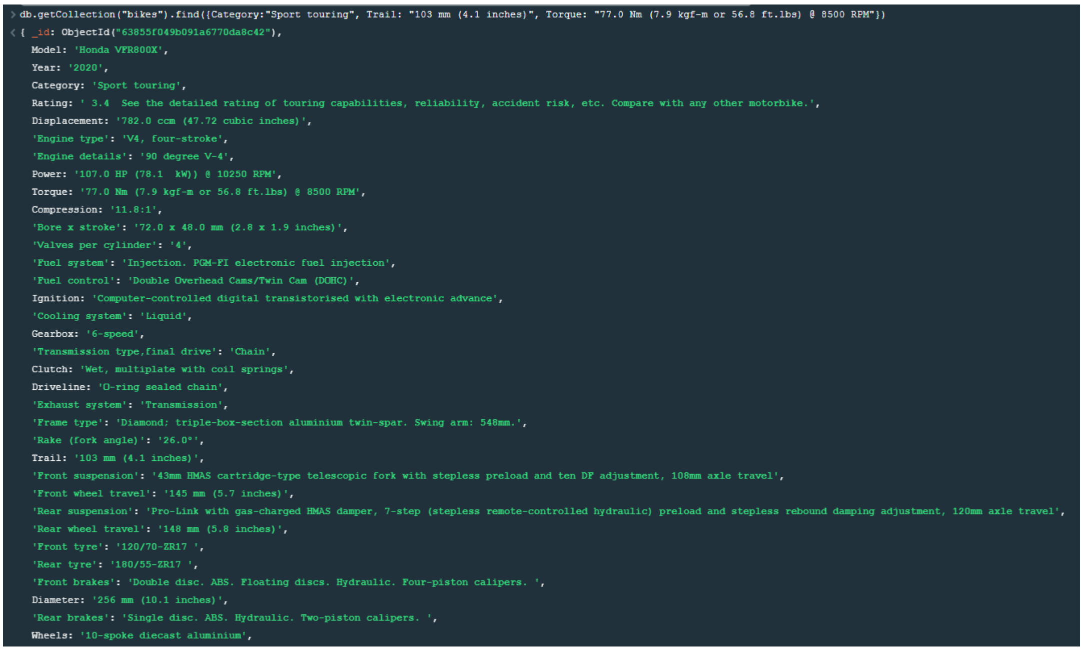
```
> db.getCollection("bikes").find({Gearbox: "100-speed",Category: "ATV", Displacement:"722.0 ccm (44.06 cubic inches)"})
< { _id: ObjectId("63855f059b091a6770da977e"),
    Model: 'Suzuki KingQuad 750AXi',
    Year: '2019',
    Category: 'ATV',
    'Price as new': 'US$ 8799.   MSRP depend on country, taxes, accessories, etc.',
    Rating: 'Do you know this bike?Click here to rate it. We miss 2 votes to show the rating.',
    Displacement: '722.0 ccm (44.06 cubic inches)',
    'Engine type': 'Single cylinder, four-stroke',
    Compression: '10.0:1',
    'Bore x stroke': '104.0 x 85.0 mm (4.1 x 3.3 inches)',
    'Valves per cylinder': '4',
    'Fuel system': 'Injection',
    'Fuel control': 'Double Overhead Cams/Twin Cam (DOHC)',
    Ignition: 'Electronic ignition',
    'Lubrication system': 'Wet sump',
    'Cooling system': 'Liquid',
    Gearbox: '100-speed',
    'Transmission type,final drive': 'Belt',
    Clutch: 'Automatic, x 2 speed with reverse and front diff-locked 4WD',
    'Front suspension': 'Independent, double wishbone, coil spring, oil damped with five-way preload adjustable shock absorbers',
    'Front wheel travel': '180 mm (7.1 inches)',
    'Rear suspension': 'Independent, double wishbone, coil spring, oil damped with five-way preload adjustable shock absorbers',
    'Rear wheel travel': '201 mm (7.9 inches)',
    'Front tyre': '25/8-12 ',
    'Rear tyre': '25/10-12 ',
    'Front brakes': 'Double disc',
    'Rear brakes': 'Sealed multidisc',
    Wheels: '2WD, 4WD  and  4WD Differential Lock',
    'Weight incl': { ' oil, gas, etc': '320.0 kg (705.5 pounds)' },
    'Seat height': '820 mm (32.3 inches) If adjustable, lowest setting.',
    'Overall height': '1285 mm (50.6 inches)',
    'Overall length': '2150 mm (84.6 inches)',
    'Overall width': '1215 mm (47.8 inches)',
```

3. db.getCollection("bikes").find({Year:"1902", Displacement: "955.0 ccm (58.27 cubic inches)"})



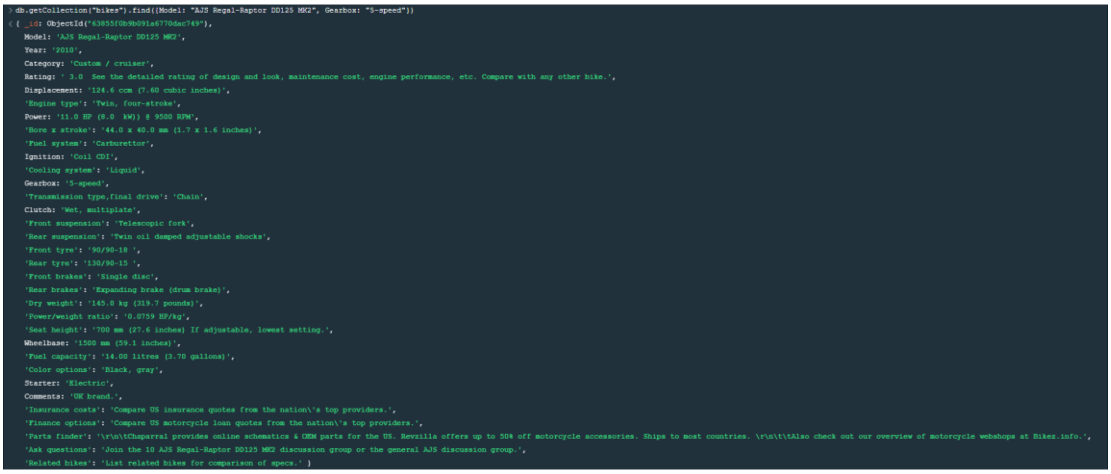
```
> db.getCollection("bikes").find({Year:"1902", Displacement: "955.0 ccm (58.27 cubic inches)"})
< { _id: ObjectId("63855f109b091a6770db1bdb"),
    Model: 'De Dion-Bouton Tricycle',
    Year: '1902',
    Category: 'Allround',
    Rating: 'Do you know this bike?Click here to rate it. We miss 2 votes to show the rating.',
    Displacement: '955.0 ccm (58.27 cubic inches)',
    'Engine type': 'Single cylinder, four-stroke',
    Power: '8.0 HP (5.8  kW)) @ 1800 RPM',
    'Top speed': '109.0 km/h (67.7 mph)',
    'Bore x stroke': '100.0 x 120.0 mm (3.9 x 4.7 inches)',
    'Fuel system': 'Carburettor. Surface carburator',
    'Cooling system': 'Air',
    Clutch: 'Direct drive  via a pair of gears from the motor directly to the rear axle',
    'Frame type': 'Decauville, steel',
    Wheels: 'Two rear wheels. Michelin pneumatic tires.',
    'Dry weight': '88.0 kg (194.0 pounds)',
    'Power/weight ratio': '0.0909 HP/kg',
    'Overall width': '920 mm (36.2 inches)',
    'Color options': 'Black',
    Comments: 'French made motorbike. French racer Georges Osmont set a speed record of 109.1 km/h in Nice with a De Dion-Bouton motor tricycle in 1892.',
    'Ask questions': 'Join the 02 De Dion-Bouton Tricycle discussion group or the general De Dion-Bouton discussion group.',
    'Related bikes': 'List related bikes for comparison of specs.' }
```

4. db.getCollection("bikes").find({Category:"Sport touring", Trail: "103 mm (4.1 inches)", Torque: "77.0 Nm (7.9 kgf-m or 56.8 ft.lbs) @ 8500 RPM"})



```
> db.getCollection("bikes").find({Category:"Sport touring", Trail: "103 mm (4.1 inches)", Torque: "77.0 Nm (7.9 kgf-m or 56.8 ft.lbs) @ 8500 RPM"})
< { _id: ObjectId("63855f049b091a6770da8c42"),
    Model: 'Honda VFR800X',
    Year: '2020',
    Category: 'Sport touring',
    Rating: ' 3.4  See the detailed rating of touring capabilities, reliability, accident risk, etc. Compare with any other motorbike.',
    Displacement: '782.0 ccm (47.72 cubic inches)',
    'Engine type': 'V4, four-stroke',
    'Engine details': '90 degree V-4',
    Power: '107.0 HP (78.1  kW)) @ 10250 RPM',
    Torque: '77.0 Nm (7.9 kgf-m or 56.8 ft.lbs) @ 8500 RPM',
    Compression: '11.8:1',
    'Bore x stroke': '72.0 x 48.0 mm (2.8 x 1.9 inches)',
    'Valves per cylinder': '4',
    'Fuel system': 'Injection. PGM-FI electronic fuel injection',
    'Fuel control': 'Double Overhead Cams/Twin Cam (DOHC)',
    Ignition: 'Computer-controlled digital transistorised with electronic advance',
    'Cooling system': 'Liquid',
    Gearbox: '6-speed',
    'Transmission type,final drive': 'Chain',
    Clutch: 'Wet, multiplate with coil springs',
    Driveline: 'O-ring sealed chain',
    'Exhaust system': 'Transmission',
    'Frame type': 'Diamond; triple-box-section aluminium twin-spar. Swing arm: 548mm.',
    'Rake (fork angle)': '26.0°',
    Trail: '103 mm (4.1 inches)',
    'Front suspension': '43mm HMAS cartridge-type telescopic fork with stepless preload and ten DF adjustment, 108mm axle travel',
    'Front wheel travel': '145 mm (5.7 inches)',
    'Rear suspension': 'Pro-Link with gas-charged HMAS damper, 7-step (stepless remote-controlled hydraulic) preload and stepless rebound damping adjustment, 120mm axle travel',
    'Rear wheel travel': '148 mm (5.8 inches)',
    'Front tyre': '120/70-ZR17 ',
    'Rear tyre': '180/55-ZR17 ',
    'Front brakes': 'Double disc. ABS. Floating discs. Hydraulic. Four-piston calipers. ',
    Diameter: '256 mm (10.1 inches)',
    'Rear brakes': 'Single disc. ABS. Hydraulic. Two-piston calipers. ',
    Wheels: '10-spoke diecast aluminium',
```

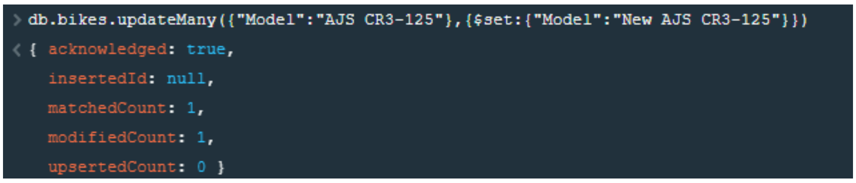
5. db.getCollection("bikes").find({Model: "AJS Regal-Raptor DD125 MK2", Gearbox: "5-speed"})



```
> db.getCollection("bikes").find({Model: "AJS Regal-Raptor DD125 MK2", Gearbox: "5-speed"})
< { _id: ObjectId("63855f0b9b091a6770dac749"),
    Model: 'AJS Regal-Raptor DD125 MK2',
    Year: '2010',
    Category: 'Custom / cruiser',
    Rating: ' 3.0  See the detailed rating of design and look, maintenance cost, engine performance, etc. Compare with any other bike.',
    Displacement: '124.6 ccm (7.60 cubic inches)',
    'Engine type': 'Twin, four-stroke',
    Power: '11.0 HP (8.0  kW)) @ 9500 RPM',
    'Bore x stroke': '44.0 x 40.0 mm (1.7 x 1.6 inches)',
    'Fuel system': 'Carburettor',
    Ignition: 'Coil CDI',
    'Cooling system': 'Liquid',
    Gearbox: '5-speed',
    'Transmission type,final drive': 'Chain',
    Clutch: 'Wet, multiplate',
    'Front suspension': 'Telescopic fork',
    'Rear suspension': 'Twin oil damped adjustable shocks',
    'Front tyre': '90/90-18 ',
    'Rear tyre': '130/90-15 ',
    'Front brakes': 'Single disc',
    'Rear brakes': 'Expanding brake (drum brake)',
    'Dry weight': '145.0 kg (319.7 pounds)',
    'Power/weight ratio': '0.0759 HP/kg',
    'Seat height': '700 mm (27.6 inches) If adjustable, lowest setting.',
    Wheelbase: '1500 mm (59.1 inches)',
    'Fuel capacity': '14.00 litres (3.70 gallons)',
    'Color options': 'Black, gray',
    Starter: 'Electric',
    Comments: 'UK brand.',
    'Insurance costs': 'Compare US insurance quotes from the nation\'s top providers.',
    'Finance options': 'Compare US motorcycle loan quotes from the nation\'s top providers.',
    'Parts finder': '\r\n\tChaparral provides online schematics & OEM parts for the US. Revzilla offers up to 50% off motorcycle accessories. Ships to most countries. \r\n\t\tAlso check out our overview of motorcycle webshops at Bikez.info.',
    'Ask questions': 'Join the 10 AJS Regal-Raptor DD125 MK2 discussion group or the general AJS discussion group.',
    'Related bikes': 'List related bikes for comparison of specs.' }
```

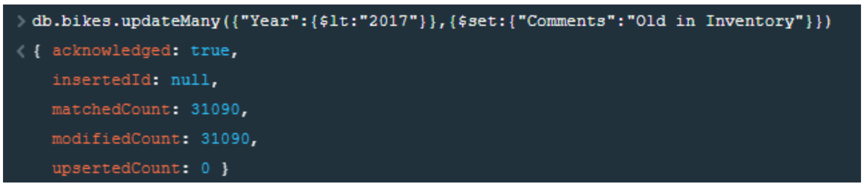
2.7 5 update queries to update some of the values in the database

1. db.bikes.updateMany({"Model":"AJS CR3-125"},{$set:{"Model":"New AJS CR3-125"}})



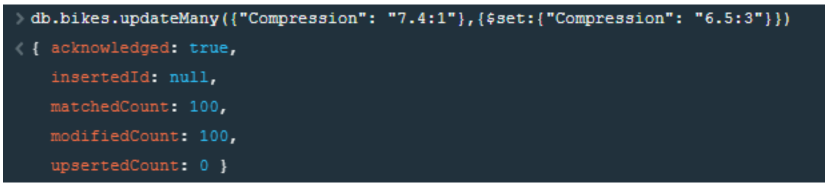
```
> db.bikes.updateMany({"Model":"AJS CR3-125"},{$set:{"Model":"New AJS CR3-125"}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
```

2. db.bikes.updateMany({"Year":{$lt:"2017"}},{$set:{"Comments":"Old in Inventory"}})



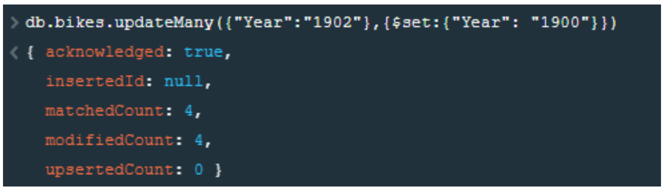
```
> db.bikes.updateMany({"Year":{$lt:"2017"}},{$set:{"Comments":"Old in Inventory"}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 31090,
    modifiedCount: 31090,
    upsertedCount: 0 }
```

3. db.bikes.updateMany({"Compression": "7.4:1"},{$set:{"Compression": "6.5:3"}})



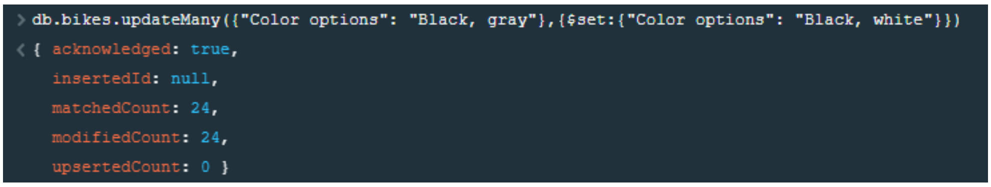
```
> db.bikes.updateMany({"Compression": "7.4:1"},{$set:{"Compression": "6.5:3"}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 100,
    modifiedCount: 100,
    upsertedCount: 0 }
```

4. db.bikes.updateMany({"Year":"1902"},{$set:{"Year": "1900"}})



```
> db.bikes.updateMany({"Year":"1902"},{$set:{"Year": "1900"}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 4,
    modifiedCount: 4,
    upsertedCount: 0 }
```

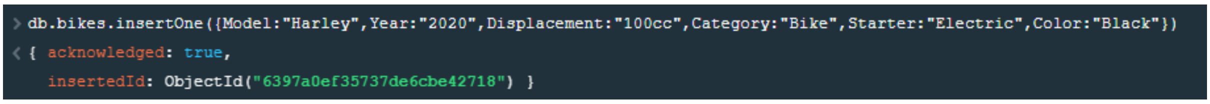
5. db.bikes.updateMany({"Color options": "Black, gray"},{$set:{"Color options": "Black, white"}})



```
> db.bikes.updateMany({"Color options": "Black, gray"},{$set:{"Color options": "Black, white"}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 24,
    modifiedCount: 24,
    upsertedCount: 0 }
```

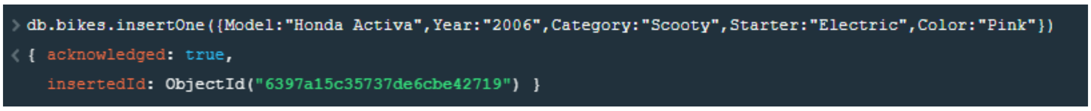
2.8 5 queries to insert new documents in the database

1. db.bikes.insertOne({Model:"Harley",Year:"2020",Displacement:"100cc",Category:"Bike",Starter:"Electric",Color:"Black"})



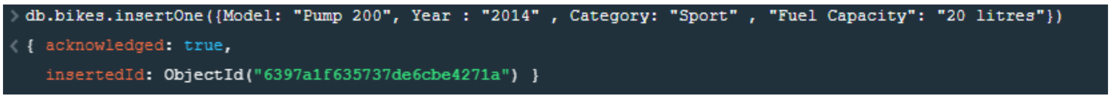
```
> db.bikes.insertOne({Model:"Harley",Year:"2020",Displacement:"100cc",Category:"Bike",Starter:"Electric",Color:"Black"})
< { acknowledged: true,
    insertedId: ObjectId("6397a0ef35737de6cbe42718") }
```

2. db.bikes.insertOne({Model:"Honda Activa",Year:"2006",Category:"Scooty",Starter:"Electric",Color:"Pink"}



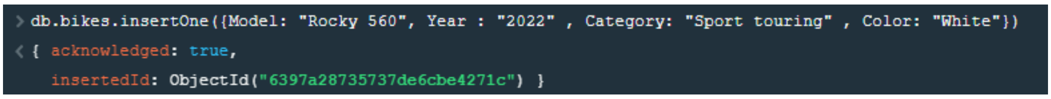
```
> db.bikes.insertOne({Model:"Honda Activa",Year:"2006",Category:"Scooty",Starter:"Electric",Color:"Pink"})
< { acknowledged: true,
    insertedId: ObjectId("6397a15c35737de6cbe42719") }
```

3. db.bikes.insertOne({Model: "Pump 200", Year : "2014" , Category: "Sport" , "Fuel Capacity": "20 litres"})



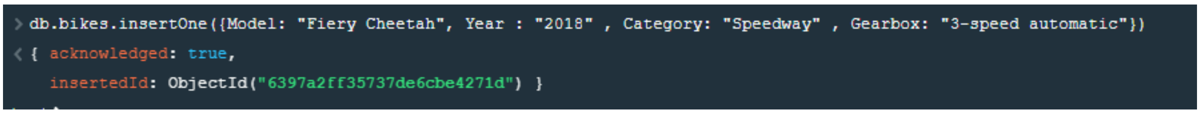
```
> db.bikes.insertOne({Model: "Pump 200", Year : "2014" , Category: "Sport" , "Fuel Capacity": "20 litres"})
< { acknowledged: true,
    insertedId: ObjectId("6397a1f635737de6cbe4271a") }
```

4. db.bikes.insertOne({Model: "Rocky 560", Year : "2022" , Category: "Sport touring" , Color: "White"})



```
> db.bikes.insertOne({Model: "Rocky 560", Year : "2022" , Category: "Sport touring" , Color: "White"})
< { acknowledged: true,
    insertedId: ObjectId("6397a28735737de6cbe4271c") }
```

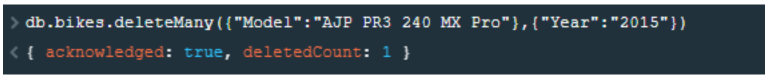
5. db.bikes.insertOne({Model: "Fiery Cheetah", Year : "2018" , Category: "Speedway" , Gearbox: "3-speed automatic"})



```
> db.bikes.insertOne({Model: "Fiery Cheetah", Year : "2018" , Category: "Speedway" , Gearbox: "3-speed automatic"})
< { acknowledged: true,
    insertedId: ObjectId("6397a2ff35737de6cbe4271d") }
```

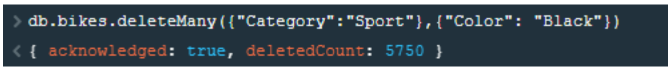
2.9 5 delete queries to remove some of the values from the database

1. db.bikes.deleteMany({"Model":"AJP PR3 240 MX Pro"},{"Year":"2015"})



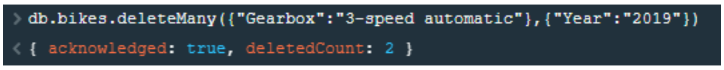
```
> db.bikes.deleteMany({"Model":"AJP PR3 240 MX Pro"},{"Year":"2015"})
< { acknowledged: true, deletedCount: 1 }
```

2. db.bikes.deleteMany({"Category":"Sport"},{"Color": "Black"})



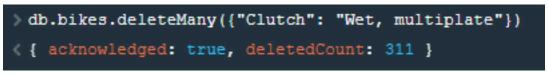
```
> db.bikes.deleteMany({"Category":"Sport"},{"Color": "Black"})
< { acknowledged: true, deletedCount: 5750 }
```

3. db.bikes.deleteMany({"Gearbox":"3-speed automatic"},{"Year":"2019"})



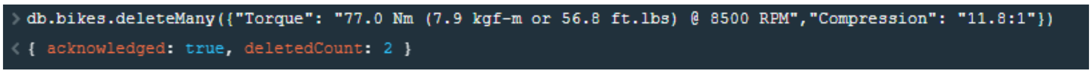
```
> db.bikes.deleteMany({"Gearbox":"3-speed automatic"},{"Year":"2019"})
< { acknowledged: true, deletedCount: 2 }
```

4. db.bikes.deleteMany({"Clutch": "Wet, multiplate"})



```
> db.bikes.deleteMany({"Clutch": "Wet, multiplate"})
< { acknowledged: true, deletedCount: 311 }
```

5. db.bikes.deleteMany({"Torque": "77.0 Nm (7.9 kgf-m or 56.8 ft.lbs) @ 8500 RPM","Compression": "11.8:1"})



```
> db.bikes.deleteMany({"Torque": "77.0 Nm (7.9 kgf-m or 56.8 ft.lbs) @ 8500 RPM","Compression": "11.8:1"})
< { acknowledged: true, deletedCount: 2 }
```

Below are the modifications made to Project 1 based on the feedback received:

We have highlighted the changes/additions in yellow for your reference.

Step 4: Design a Database

Schema Design

Entity: customer

Primary key: customer_id
Foreign key: birth_year

Entity: generations

Primary key: birth_year

Entity: product

Primary key: product_id

Entity: staff

Primary key: staff_id

Entity: pastryinventory

Primary key: sales_outlet_id
Foreign key: product_id

Entity: salesreceipts

Primary key: transaction_id
Foreign key: sales_outlet_id, staff_id

Entity: sales_outlet

Primary key: sales_outlet_id

Entity: salestargets

Primary key: sales_outlet_id

Entity: dates

Primary key: transaction_date

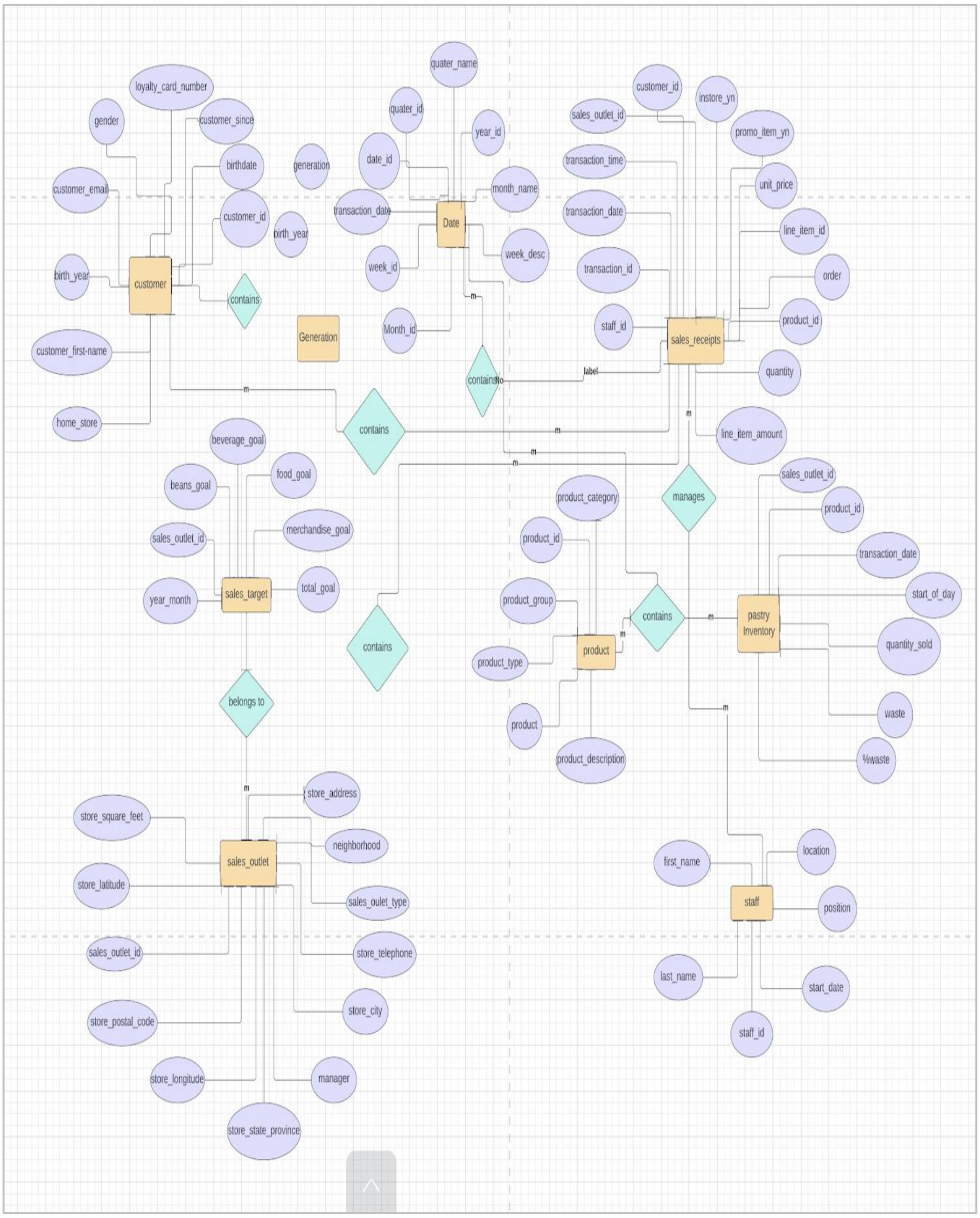
Relationships:

1. customer to generation:
   Each customer belongs to one Generation (Baby Boomers, GenX, GenZ, Older Millenials, and Younger Millenials)
2. salesreceipts to Customer:
   Each sales receipt corresponds to one customer
3. salesreceipts to dates:
   Each sales receipt can have only one date
4. salesreceipts to staff:
   Each sales receipt corresponds to one staff member
5. salesreceipts to sales_outlet:
   Each sales receipt matches with a transaction that occurred at one sales outlet
6. sales_outlet to salestargets:
   Each sales outlet is allocated a sales target
7. pastryinventory to product:

Each row in the pastry inventory table can correspond to one or many product(s)

8. pastryinventory to dates:
   Each inventory entry can have only one date

ER Diagram:



Schema Normalization:

Functional Dependencies from the schema

- customer table
  - customer_id **->** **{** customer_name, loyalty_card_number **}**
- product table
  - product_id **->** **{** product_group, product_category, product_type **}**
- sales_outlet table

    - sales_outlet_id **->** **{** sales_outlet_type, store_address, store_city, Neighborhood **}**
- staff table
    - staff_id **->** **{** first_name, last_name, position, location **}**
- salesreceipts table
    - transaction_id **->** **{** transaction_date, transaction_time, order, quantity **}**
- pastryinventory table
    - sales_outlet_id **->** **{** transaction_date, product_id, start_of_day, quantity_sold, waste **}**
- dates table
    - transaction_date **->** **{** Date_id, Week_id, Month_id, Year_id, Quarter_id **}**
- salestarget table
    - sales_outlet_id **->** **{** year_month, total_goal **}**
- generations table
    - generation **->** **{** birth_year **}**

Check if your schema is in BCNF (Boyce-Codd Normal Form)

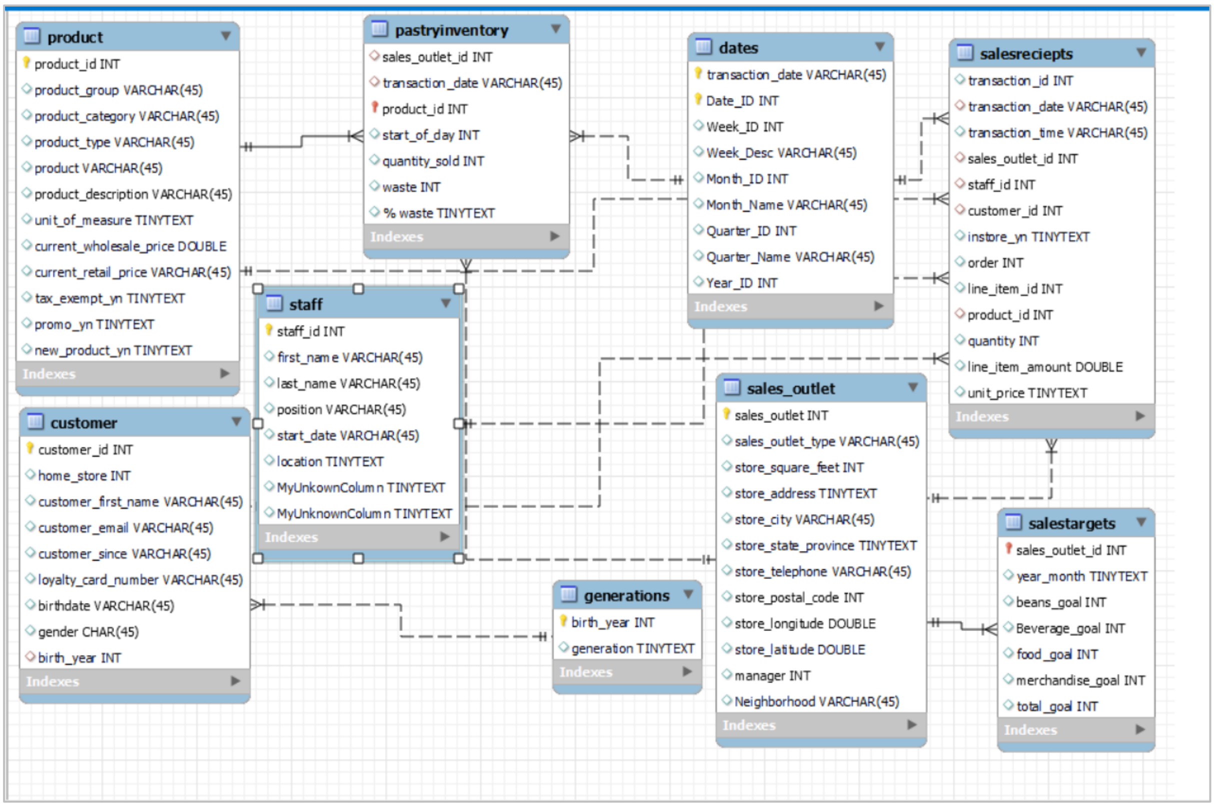
A schema is in BCNF, if a table is in 3NF and has the table's super key for each functional dependency.
In our dataset, there is a super key for each functional dependency to verify that the schema is in BCNF, such as

- customer id, customer name, and loyalty card number
- product id, product group, product category, and product type
- sales outlet id, sales outlet type, store address, store city, and neighborhood
- staff id, first name, last name, position, and location.
- transaction id, transaction date, transaction time, order, and quantity
- sales outlet id, transaction date, product id, start of day, quantity sold, waste
- transaction date, date id, week id, month id, year id, quarter id
- sales outlet id, year & month, total goal
- generation and birth year

Every functional dependence has a super key, which demonstrates that the schema is in BCNF.

There were no errors while importing data into the database

Schema Design



Step 5: Data Cleaning and Database Testing

A. Statistics

We have carried out the basic statistical analysis including range, mean, variance, and frequency in Step 3. Some of the inferences we can make based on it are:

a. From the 'salesreceipts' table, we can infer that the sales for the month of April 2019 are being analyzed, based on the range of the column 'transaction_date'. The column line_item_amount (net selling price of an order), where the mean is $4.68, implies that an order placed by the customer in these coffee shops, on average amount to $4.68.

b. From the 'customer' table, we can see that the customers who visited the coffee shop in the given time period were born between 1950 and 2001.

c. From the 'pastryinventory' we can say that inventory value at the start of the day on average is $24.06 with a high variability over the month, the average quantity sold is $9.3, and the % waste was around 58%.

d. The product table shows that the average current_retail_price is 69% higher than the average current_wholesale_price. (% change calculated from $3.89 to $6.58)

e. On average, each sales outlet is expected to meet a sales target of $19437.5, including beans, beverages, food, and merchandise.

B. General Queries for importing and sorting the dataset

The queries used are as follows:

1. 201904 sales receipts (Salesreciepts) –
   i. Alter table name from **201904 sales receipts** to **Salesreciepts** –
      ALTER TABLE `Project1`.`201904 sales receipts`
      RENAME TO `Project1`.`Salesreciepts`

ii. Alter **transaction date** datatype from **TEXT** to **DATE**

```
ALTER TABLE `Project1`.`Salesreciepts`
CHANGE COLUMN `transaction_date` `transaction_date`
DATE NULL DEFAULT NULL
```

iii. Set **transaction_id** as a primary key

```
ALTER TABLE `project`.`salesreciepts`
CHANGE COLUMN `transaction_id` `transaction_id` INT NOT NULL ,
ADD PRIMARY KEY (`transaction_id`)
```

2. customer (Customer)

i. Set **customer id** as a **primary key**

```
ALTER TABLE `Project1`.`customer`
CHANGE COLUMN `customer_id` `customer_id` INT NOT NULL,
ADD PRIMARY KEY (`customer_id`)
```

3. Generations

i. Set **birth_year** as a **primary key**

```
ALTER TABLE `Project1`.`generations`
CHANGE COLUMN `birth_year` `birth_year` INT NOT NULL,
ADD PRIMARY KEY (`birth_year`)
```

4. Pastry inventory

i. ALTER TABLE `project`.`pastry inventory`

```
ALTER TABLE `project`.`pastry inventory`
ADD CONSTRAINT `sales_outlet_idFK`
FOREIGN KEY (`sales_outlet_id`)
REFERENCES `project`.`salesoutlet` (`sales_outlet_id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
ADD CONSTRAINT `product_idFK`
FOREIGN KEY (`product_id`) REFERENCES `project`.`product` (`product_id`)
```

5. Product

i. Set product_id as a primary key

```
ALTER TABLE `Project1`.`product`
CHANGE COLUMN `product_id` `product_id` INT NOT NULL,
ADD PRIMARY KEY (`product_id`)
```

6. Sales_outlet

i. Set sales_outlet_id as a primary key

```
ALTER TABLE `Project1`.`sales_outlet`
CHANGE COLUMN `sales_outlet_id` `sales_outlet_id` INT NOT NULL,
ADD PRIMARY KEY (`sales_outlet_id`)
```

ii. Add missing figure **'30' under column manager where row sales_out_id is 2**

```
UPDATE `Project1`.`sales_outlet` SET `manager` = '30'
WHERE (`sales_outlet_id` = '2')
```

7. Sales target (Salestargets)

   i. Set sales_outlet_id as a primary key and table name to Salestargets

```
ALTER TABLE `Project1`.`sales targets`
CHANGE COLUMN `sales_outlet_id` `sales_outlet_id` INT NOT NULL,
ADD PRIMARY KEY (`sales_outlet_id`); , RENAME TO `Project1`.`Sales_targets`
```

8. Staff

   i. Set staff_id as a primary key

```
ALTER TABLE `Project1`.`staff`
CHANGE COLUMN `staff_id` `staff_id` INT NOT NULL,
ADD PRIMARY KEY (`staff_id`)
```

   ii. Delete 2 unknown column

```
ALTER TABLE `Project1`.`staff`
DROP COLUMN `MyUnknownColumn_[0]`,
DROP COLUMN `MyUnknownColumn
```

9. Dates

   i. Alter transaction date datatype from **TEXT** to **DATE**

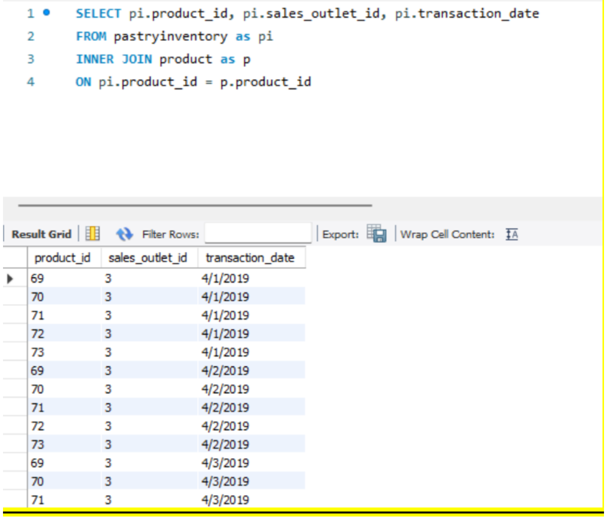
```
ALTER TABLE `Project1`.`dates`
CHANGE COLUMN `transaction_date` `transaction_date` DATE NULL DEFAULT NULL
```

   ii. Set transaction_date as a primary key

```
ALTER TABLE `Project1`.`dates`
CHANGE COLUMN `transaction_date` `transaction_date` DATE NOT NULL,
ADD PRIMARY KEY (`transaction_date`)
```

Join Queries to test the database

```
SELECT pi.product_id, pi.sales_outlet_id, pi.transaction_date
FROM pastryinventory as pi
INNER JOIN product as p
ON pi.product_id = p.product_id
```



```
SELECT pi.product_id, pi.sales_outlet_id, pi.transaction_date
FROM pastryinventory as pi
INNER JOIN product as p
ON pi.product_id = p.product_id
```

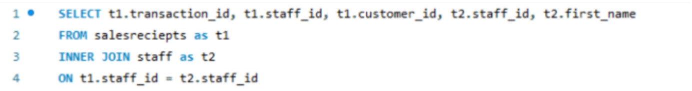
product_id	sales_outlet_id	transaction_date
69	3	4/1/2019
70	3	4/1/2019
71	3	4/1/2019
72	3	4/1/2019
73	3	4/1/2019
69	3	4/2/2019
70	3	4/2/2019
71	3	4/2/2019
72	3	4/2/2019
73	3	4/2/2019
69	3	4/3/2019
70	3	4/3/2019
71	3	4/3/2019

SELECT t1.transaction_id, t1.staff_id, t1.customer_id, t2.staff_id, t2.first_name

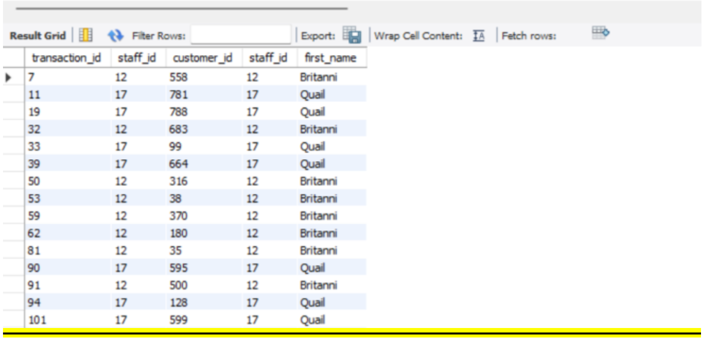
FROM salesreciepts as t1

INNER JOIN staff as t2

ON t1.staff_id = t2.staff_id



```
SELECT t1.transaction_id, t1.staff_id, t1.customer_id, t2.staff_id, t2.first_name
FROM salesreciepts as t1
INNER JOIN staff as t2
ON t1.staff_id = t2.staff_id
```



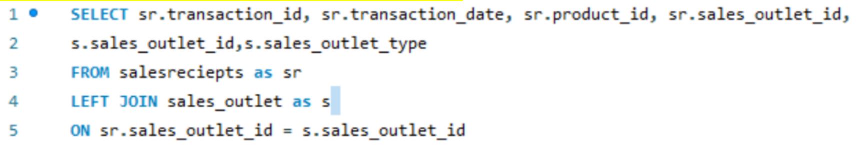
transaction_id	staff_id	customer_id	staff_id	first_name
7	12	558	12	Britanni
11	17	781	17	Quail
19	17	788	17	Quail
32	12	683	12	Britanni
33	17	99	17	Quail
39	17	664	17	Quail
50	12	316	12	Britanni
53	12	38	12	Britanni
59	12	370	12	Britanni
62	12	180	12	Britanni
81	12	35	12	Britanni
90	17	595	17	Quail
91	12	500	12	Britanni
94	17	128	17	Quail
101	17	599	17	Quail

SELECT sr.transaction_id, sr.transaction_date, sr.product_id, sr.sales_outlet_id, s.sales_outlet_id,s.sales_outlet_type

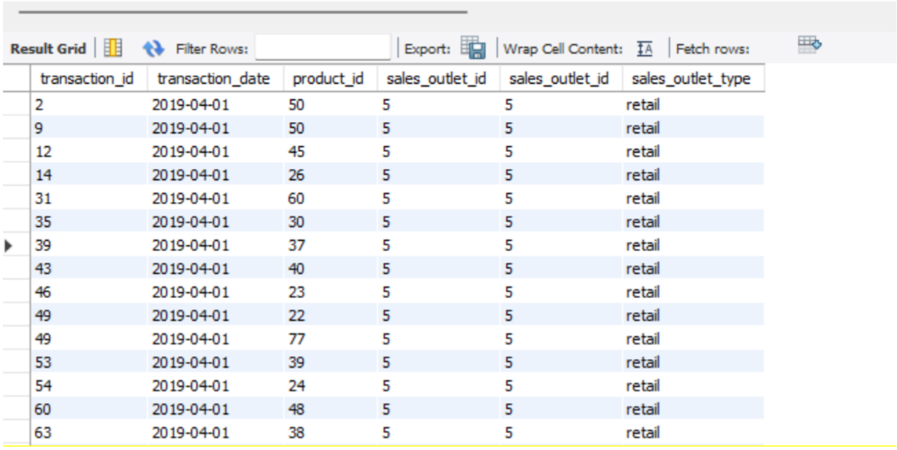
FROM salesreciepts as sr

LEFT JOIN sales_outlet as s

ON sr.sales_outlet_id = s.sales_outlet_id



```
SELECT sr.transaction_id, sr.transaction_date, sr.product_id, sr.sales_outlet_id,
s.sales_outlet_id,s.sales_outlet_type
FROM salesreciepts as sr
LEFT JOIN sales_outlet as s
ON sr.sales_outlet_id = s.sales_outlet_id
```



transaction_id	transaction_date	product_id	sales_outlet_id	sales_outlet_id	sales_outlet_type
2	2019-04-01	50	5	5	retail
9	2019-04-01	50	5	5	retail
12	2019-04-01	45	5	5	retail
14	2019-04-01	26	5	5	retail
31	2019-04-01	60	5	5	retail
35	2019-04-01	30	5	5	retail
39	2019-04-01	37	5	5	retail
43	2019-04-01	40	5	5	retail
46	2019-04-01	23	5	5	retail
49	2019-04-01	22	5	5	retail
49	2019-04-01	77	5	5	retail
53	2019-04-01	39	5	5	retail
54	2019-04-01	24	5	5	retail
60	2019-04-01	48	5	5	retail
63	2019-04-01	38	5	5	retail

Conclusion

Using SQL, we were able to observe, clean, and manipulate the retail data of a coffee chain to generate valuable insights and inferences. The dataset required minimal cleansing which we were able to do with commands like CREATE, ALTER, and DROP. The overall data quality was good. We were able to understand the data better by computing the basic statistical measures like mean, std dev, and variance. However, for a more comprehensive and exhaustive analysis of the business, we would need more data points like sales data across all months. Additionally, better clarity on some of the columns will help ensure more appropriate business decisions.