DDL Commands for creating the tables:

```
CREATE TABLE Customers (customerID int,
                        name varchar(100),
                        PRIMARY_KEY (customerID) ;

CREATE TABLE Include (purchaseID varchar(15),
                      productID varchar(15),
                      storeName int,
                      PRIMARY_KEY(purchaseID, productID, storeName),
                      FOREIGN_KEY (purchaseID) REFERENCES
                      Purchases(purchaseID),
                      FOREIGN_KEY (productID) REFERENCES Products(productID),
                      FOREIGN_KEY(storeName) REFERENCES Stores(storeName) ;

CREATE TABLE Products (productID varchar(100),
                       storeName varchar(100),
                       item varchar(100),
                       price  double(10,2)
                       PRIMARY_KEY(productID)) ;

CREATE TABLE Purchases (purchaseID int,
                        customerID int,
                        date datetime,
                        PRIMARY_KEY(purchaseID, customerID),
                        FOREIGN_KEY(purchaseID) REFERENCES
                        Purchases(purchaseID),
                        FOREIGN_KEY(customerID) REFERENCES Customers
                        (customerID)) ;

CREATE TABLE Stores (storeName integer, PRIMARY_KEY(storeName)) ;
```

```
mysql> SELECT COUNT(*) FROM Customers;
+----------+
| COUNT(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.00 sec)
```

```
mysql> SELECT COUNT(*) FROM Include;
+----------+
| COUNT(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.00 sec)
```

```
mysql> SELECT COUNT(*) FROM Products;
+----------+
| COUNT(*) |
+----------+
|    14768 |
+----------+
1 row in set (0.00 sec)
```

```
mysql> SELECT COUNT(*) FROM Purchases;
+----------+
| COUNT(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.00 sec)
```

```
mysql> SELECT COUNT(*) FROM Stores;
+----------+
| COUNT(*) |
+----------+
|      924 |
+----------+
1 row in set (0.00 sec)
```

customerID	name
1	Albert Alice Debra
2	Albert Alice Sanchez
3	Albert Annie Gerald
4	Albert Anthony Alexander
5	Albert Anthony Jackson
6	Albert Anthony Wanda
7	Albert Bailey Stephen
8	Albert Baker Carlos
9	Albert Baker Jackson
10	Albert Baker Jacqueline
11	Albert Baker Jean
12	Albert Baker Nelson
13	Albert Brooks Ruth
14	Albert Brooks Thomas
15	Albert Chris Diaz

storeName
0
1
10
100
101
102
103
104
105
106
107
108
109
11
110

purchaseID	productID	storeName
2	DRB01	1
6	DRB01	1
8	DRB01	1
10	DRB01	1
13	DRB01	1
21	DRB01	1
55	DRB01	1
84	DRB01	1
101	DRB01	1
121	DRB01	1
125	DRB01	1
143	DRB01	1
148	DRB01	1
158	DRB01	1
163	DRB01	1

productID	storeName	item	price
DRB01	1	Soft Drinks	7.39
DRB01	10	Soft Drinks	7.39
DRB01	100	Soft Drinks	7.39
DRB01	101	Soft Drinks	7.39
DRB01	102	Soft Drinks	7.39
DRB01	103	Soft Drinks	7.39
DRB01	104	Soft Drinks	7.39
DRB01	105	Soft Drinks	7.39
DRB01	106	Soft Drinks	7.39
DRB01	107	Soft Drinks	7.39
DRB01	108	Soft Drinks	7.39
DRB01	109	Soft Drinks	7.39
DRB01	11	Soft Drinks	7.39
DRB01	110	Soft Drinks	7.39
DRB01	111	Soft Drinks	7.39

purchaseID	customerID	date
0	951	2021-08-24 11:17:43
1	257	2021-08-21 13:13:25
2	15	2021-08-06 09:43:19
3	775	2021-08-16 19:00:51
4	925	2021-08-23 19:12:23
5	791	2021-08-17 14:20:49
6	13	2021-08-31 17:44:49
7	872	2021-08-25 11:42:01
8	40	2021-08-31 07:33:19
9	689	2021-08-07 16:55:57
10	72	2021-08-23 13:16:54
11	201	2021-08-20 19:16:36
12	103	2021-08-27 16:39:11
13	67	2021-08-14 10:59:42
14	288	2021-08-21 18:27:15

Advanced Query #1:

```
SELECT purchaseID, Customers.customerID, Customers.name
FROM Purchases
JOIN Customers ON Purchases.purchaseID = Customers.customerID
WHERE name like "A%" ;
```

```
mysql> select purchaseID, Customers.customerID, Customers.name from Purchases join Customers on Purchases.purchaseID = Customers.c
ustomerID where name like "A%"
    ->
    -> limit 15;
+------------+------------+--------------------------+
| purchaseID | customerID | name                     |
+------------+------------+--------------------------+
|          1 |          1 | Albert Alice Debra       |
|          2 |          2 | Albert Alice Sanchez     |
|          3 |          3 | Albert Annie Gerald      |
|          4 |          4 | Albert Anthony Alexander |
|          5 |          5 | Albert Anthony Jackson   |
|          6 |          6 | Albert Anthony Wanda     |
|          7 |          7 | Albert Bailey Stephen    |
|          8 |          8 | Albert Baker Carlos      |
|          9 |          9 | Albert Baker Jackson     |
|         10 |         10 | Albert Baker Jacqueline  |
|         11 |         11 | Albert Baker Jean        |
|         12 |         12 | Albert Baker Nelson      |
|         13 |         13 | Albert Brooks Ruth       |
|         14 |         14 | Albert Brooks Thomas     |
|         15 |         15 | Albert Chris Diaz        |
+------------+------------+--------------------------+
15 rows in set (0.00 sec)
```

Advanced Query #2:

```
SELECT *
FROM (SELECT purchaseID
        FROM Purchases limit 100) s,
        (SELECT productID, storeName
        FROM Products LIMIT 10) t ;
```

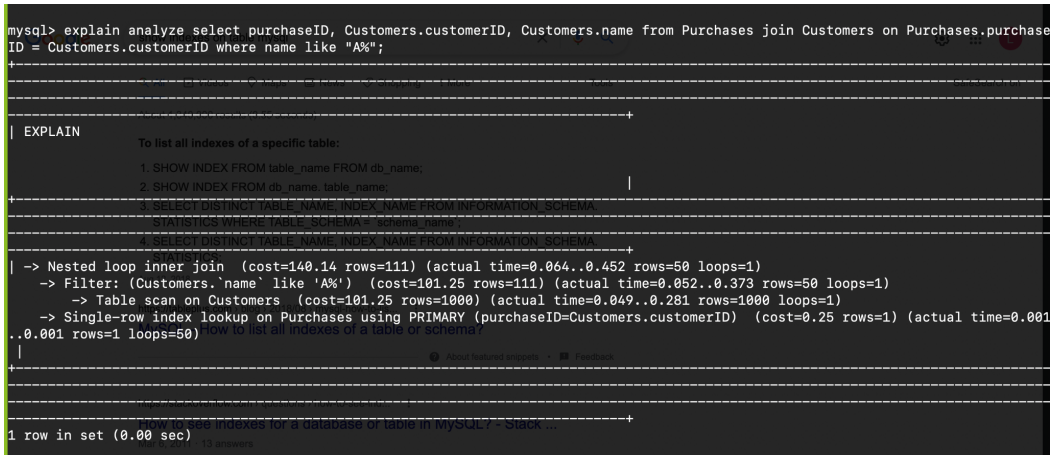
```
mysql> select * from (select purchaseID from Purchases limit 100) s, (select productID, storeName from Products limit 10) t limit
15;
+------------+-----------+-----------+
| purchaseID | productID | storeName |
+------------+-----------+-----------+
|        912 | FDU35     | 1         |
|        912 | FDU33     | 1         |
|        912 | FDU16     | 1         |
|        912 | FDQ49     | 1         |
|        912 | FDM14     | 1         |
|        912 | FDF33     | 1         |
|        912 | FDD08     | 1         |
|        912 | FDC59     | 1         |
|        912 | DRP47     | 1         |
|        912 | DRB01     | 1         |
|        341 | FDU35     | 1         |
|        341 | FDU33     | 1         |
|        341 | FDU16     | 1         |
|        341 | FDQ49     | 1         |
|        341 | FDM14     | 1         |
+------------+-----------+-----------+
15 rows in set (0.00 sec)

mysql>
```

Indexing Analysis:

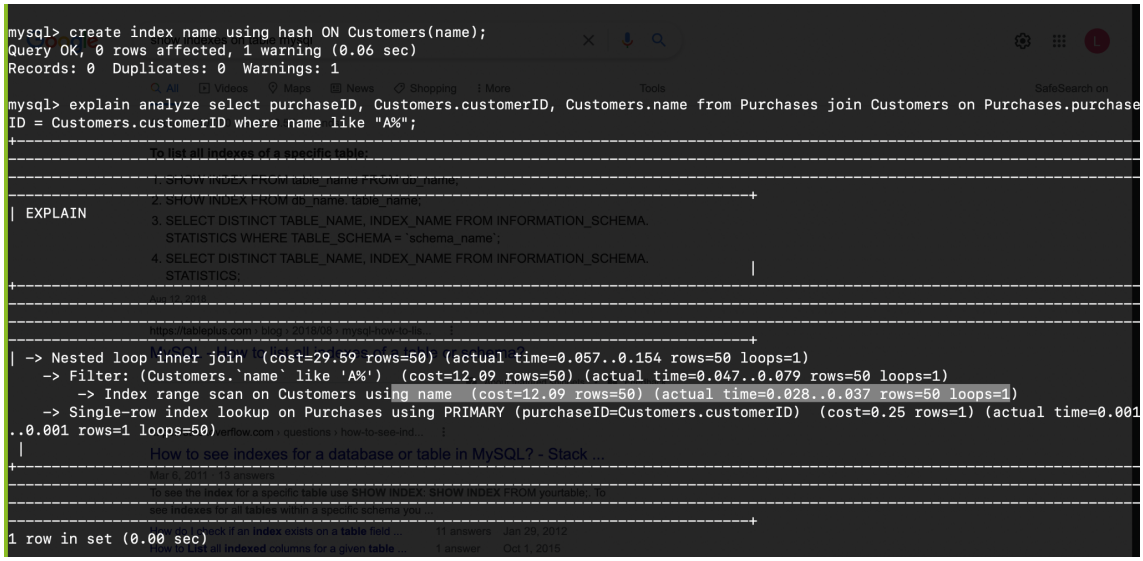
First Advanced Query:

1. Create index **purchases** on the Primary Key, Customers.customerID



Time it takes to finish scanning: 0.373-0.052=0.321 seconds

2. Create index **name** on Customers.name (USING hash)



Time it takes to finish scanning: 0.037-0.028=0.009 seconds

3. Create index **names** on Customers.name (USING btree)

```
mysql> create index names using btree on Customers (customerID);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze select purchaseID, Customers.customerID, Customers.name from Purchases join Customers on Purchases.purc
ID = Customers.customerID where name like "A%";
+-----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------+
| EXPLAIN

+-----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=29.59 rows=50) (actual time=0.035..0.093 rows=50 loops=1)
    -> Filter: (Customers.`name` like 'A%')  (cost=12.09 rows=50) (actual time=0.026..0.040 rows=50 loops=1)
        -> Index range scan on Customers using name  (cost=12.09 rows=50) (actual time=0.022..0.031 rows=50 loops=1)
    -> Single-row index lookup on Purchases using PRIMARY (purchaseID=Customers.customerID)  (cost=0.25 rows=1) (actual time=0
..0.001 rows=1 loops=50)
 |
+-----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------+

1 row in set (0.00 sec)
```

Time it takes to finish scanning: 0.031-0.022=0.009 seconds

Second Advanced Query:

1. Create index **Test1** on Primary Key, Customers.customerID

```
mysql> explain analyze select * from (select purchaseID from Purchases limit 100
) s, (select productID, storeName from Products limit 10) t
    -> ;
+-----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------+
| EXPLAIN

+-----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------+
| -> Inner hash join (no condition)  (cost=1627.40 rows=1000) (actual time=0.113..0.188 rows=1000 loops=1)
    -> Table scan on s  (cost=0.04..3.75 rows=100) (actual time=0.000..0.003 rows=100 loops=1)
        -> Materialize  (cost=111.04..114.75 rows=100) (actual time=0.039..0.046 rows=100 loops=1)
            -> Limit: 100 row(s)  (cost=101.00 rows=100) (actual time=0.017..0.028 rows=100 loops=1)
                -> Index scan on Purchases using Customer_Id  (cost=101.00 rows=1000) (actual time=0.017..0.023 rows=100 loops=1)
    -> Hash
        -> Table scan on t  (cost=0.26..2.62 rows=10) (actual time=0.001..0.002 rows=10 loops=1)
            -> Materialize  (cost=1521.18..1523.54 rows=10) (actual time=0.062..0.063 rows=10 loops=1)
```

Time it takes to finish scanning: 0.023-0.017=0.006 seconds

2. Create index **Test1** on Purchases.purchaseID (USING hash)

```
mysql> create unique index Test1 using hash ON Purchases (purchaseID);
Query OK, 0 rows affected, 1 warning (0.10 sec)
Records: 0  Duplicates: 0  Warnings: 1
```

```
mysql> explain analyze select * from (select purchaseID from Purchases limit 100
    -> ) s, (select productID, storeName from Products limit 10) t;
+------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
--------------------------------------------------------------------+
| EXPLAIN

+------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
--------------------------------------------------------------------+
| -> Inner hash join (no condition)  (cost=1627.40 rows=1000) (actual time=0.121..0.195 rows=1000 loops=1)
    -> Table scan on s  (cost=0.04..3.75 rows=100) (actual time=0.000..0.004 rows=100 loops=1)
        -> Materialize  (cost=111.04..114.75 rows=100) (actual time=0.034..0.042 rows=100 loops=1)
            -> Limit: 100 row(s)  (cost=101.00 rows=100) (actual time=0.014..0.025 rows=100 loops=1)
                -> Index scan on Purchases using Test1  (cost=101.00 rows=1000) (actual time=0.014..0.020 rows=100 loops=1)
    -> Hash
        -> Table scan on t  (cost=0.26..2.62 rows=10) (actual time=0.001..0.002 rows=10 loops=1)
            -> Materialize  (cost=1521.18..1523.54 rows=10) (actual time=0.074..0.075 rows=10 loops=1)
                -> Limit: 10 row(s)  (cost=1519.91 rows=10) (actual time=0.054..0.056 rows=10 loops=1)
```

Time it takes to finish scanning: 0.006 seconds

3. Create index **Test1** on Purchases.purchaseID (USING btree)

```
mysql> explain analyze select * from (select purchaseID from Purchases limit 100
) s, (select productID, storeName from Products limit 100) t;
+------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------+
| EXPLAIN
                                                                              |
+------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------+
| -> Inner hash join (no condition)  (cost=1118.51 rows=10000) (actual time=0.213..0.706 rows=10000 loops=1)
    -> Table scan on t  (cost=0.04..3.75 rows=100) (actual time=0.001..0.007 rows=100 loops=1)
        -> Materialize  (cost=1475.39..1479.10 rows=100) (actual time=0.083..0.095 rows=100 loops=1)
            -> Limit: 100 row(s)  (cost=1465.35 rows=100) (actual time=0.038..0.052 rows=100 loops=1)
                -> Index scan on Products using storeName  (cost=1465.35 rows=14411) (actual time=0.038..0.046 rows=100 loops=1)
    -> Hash
        -> Table scan on s  (cost=0.04..3.75 rows=100) (actual time=0.001..0.004 rows=100 loops=1)
            -> Materialize  (cost=111.04..114.75 rows=100) (actual time=0.107..0.114 rows=100 loops=1)
                -> Limit: 100 row(s)  (cost=101.00 rows=100) (actual time=0.036..0.048 rows=100 loops=1)
                    -> Index scan on Purchases using ttt  (cost=101.00 rows=1000) (actual time=0.035..0.042 rows=100 loops=1)
 |
```

Time it takes to finish scanning: 0.007 seconds

Report for advanced queries:

For the first advanced query, to find the best index, we first used the default index, which created an index based on the primary key, and found that it took 0.321 seconds to complete. Next, we created the index based on Customers.name using hash, and found that it only took 0.009 seconds to execute the query, which is an improvement. Lastly, we created an index based on Customers.name again, but used b-tree instead. It took the same amount of time to finish as using a hash for Customers.name. Either one of the later two cases will be our choice.

For the second advanced query, to find the best index, we first tried to use the default index, and found that it took 0.006 seconds to terminate. Then, we created the index based on Purchases.purchaseID using hash, and found that it took 0.006 seconds to execute the query, which is the same as the previous case. Lastly, we created an index based on Purchases.purchaseID again, but we used b-tree this time, and it came out that it took 0.007 seconds which is a little bit longer. Either one of the first two cases will be our choice.