Command:

Use salesordersexample2;

SHOW indexes in Vendors;

```
1 • Use salesordersexample2;
2 • SHOW indexes in Vendors;
3
```

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

| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| vendors | 0 | PRIMARY | 1 | VendorID | A | 10 | NULL | NULL | | BTREE | | |
| vendors | 1 | VendZipCode | 1 | VendZipCode | A | 10 | NULL | NULL | YES | BTREE | | |

(It has indexes in Column VendorID and VendZipCode)

```
3 EXPLAIN EXTENDED
4 SELECT * FROM Vendors WHERE VendZipCode=98001;
```

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

| id | select_type | table | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | SIMPLE | Vendors | ALL | VendZipCode | NULL | NULL | NULL | 10 | 100.00 | Using where |

It is following what I had expected.

That's because I had thought that it would look though all the datas in table vendors and then would check the foreign key vendorzipcodes to see which one matches among the 10 different vendors.

Here you can see from the output that a simple query had been run on the vendors table, going though the table sequentially. Although it realized that an index of VendZipcode was present, it didn't use it as an index and instead searched though all the 10 rows. Here the extra condition was where and using this, the DBMS realized it needs see 100% of the rows. Also the DBMS believes it needs to see 10 rows which is basically all the rows in that table.