1. 插入orders表的插入方式操作描述，时间截图。
2. 插入products表的插入方式操作描述，时间截图。

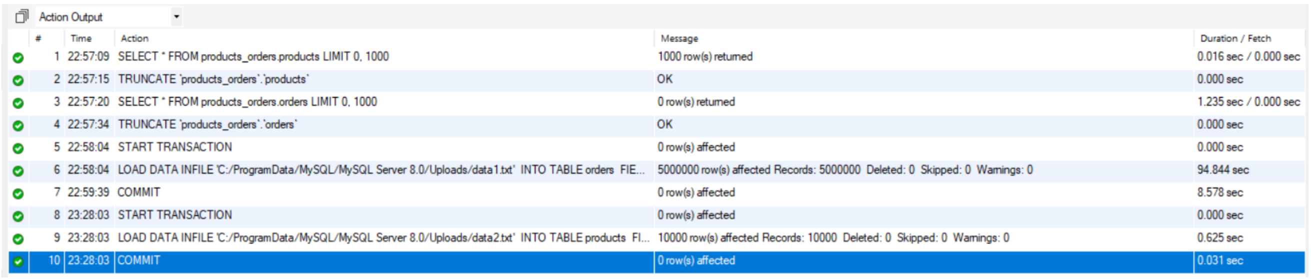
   ```
   START TRANSACTION;
   LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/data1.txt'
   INTO TABLE orders
   FIELDS TERMINATED BY '\t'
   LINES TERMINATED BY '\r\n';
   COMMIT;
   ```

   在MySQL Workbench中直接利用LOAD DATA INFILE语句插入data1.txt中数据

   ```
   START TRANSACTION;
   LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/data2.txt'
   INTO TABLE products
   FIELDS TERMINATED BY '\t'
   LINES TERMINATED BY '\r\n'
   IGNORE 0 LINES;
   COMMIT;
   ```

   在MySQL Workbench中直接利用LOAD DATA INFILE语句插入data2.txt中数据

操作及对应时间截图:

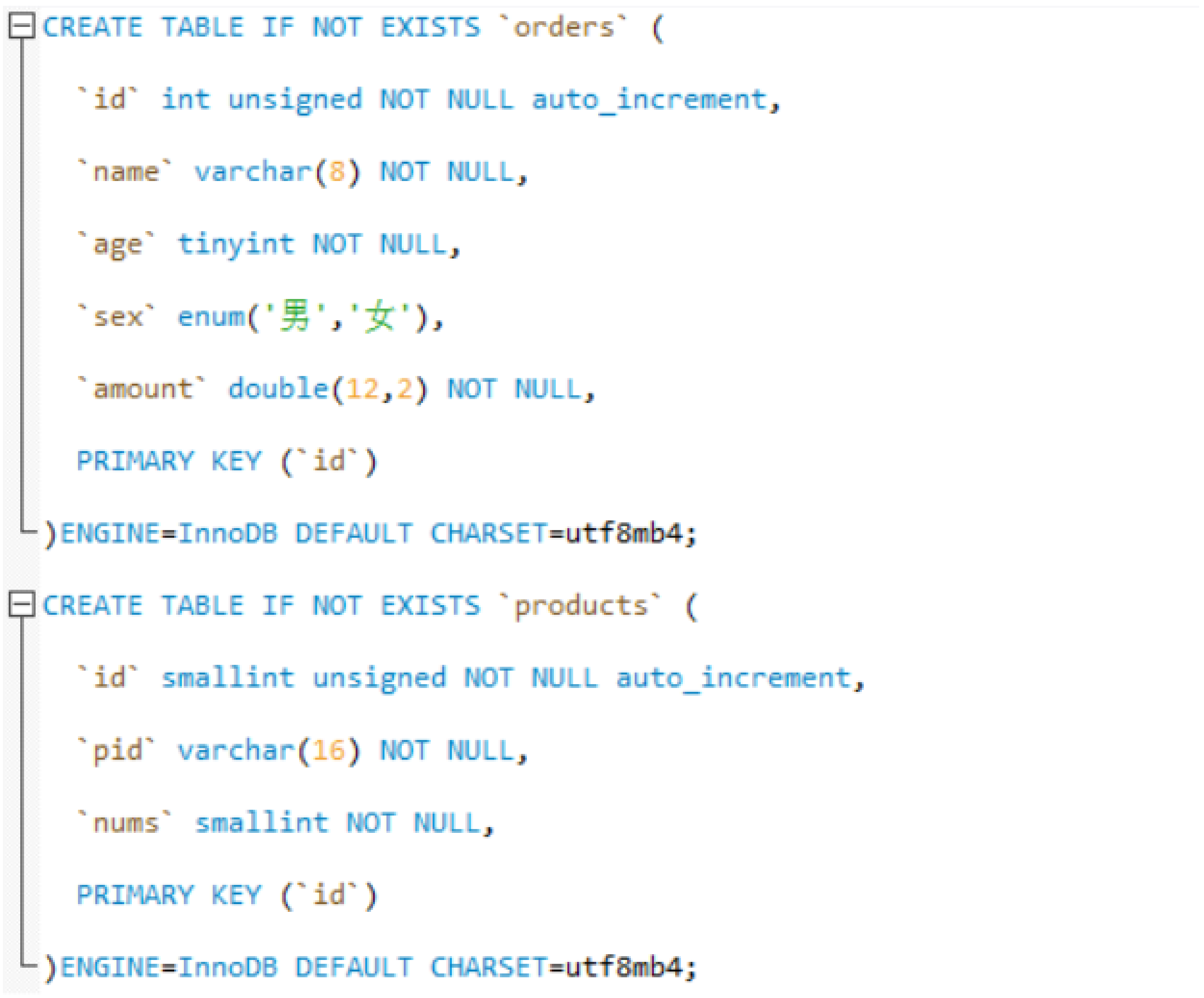


Action Output

	#	Time	Action	Message	Duration / Fetch
✓	1	22:57:09	SELECT * FROM products_orders.products LIMIT 0, 1000	1000 row(s) returned	0.016 sec / 0.000 sec
✓	2	22:57:15	TRUNCATE `products_orders`.`products`	OK	0.000 sec
✓	3	22:57:20	SELECT * FROM products_orders.orders LIMIT 0, 1000	0 row(s) returned	1.235 sec / 0.000 sec
✓	4	22:57:34	TRUNCATE `products_orders`.`orders`	OK	0.000 sec
✓	5	22:58:04	START TRANSACTION	0 row(s) affected	0.000 sec
✓	6	22:58:04	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/data1.txt' INTO TABLE orders FIE...	5000000 row(s) affected Records: 5000000 Deleted: 0 Skipped: 0 Warnings: 0	94.844 sec
✓	7	22:59:39	COMMIT	0 row(s) affected	8.578 sec
✓	8	23:28:03	START TRANSACTION	0 row(s) affected	0.000 sec
✓	9	23:28:03	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/data2.txt' INTO TABLE products FI...	10000 row(s) affected Records: 10000 Deleted: 0 Skipped: 0 Warnings: 0	0.625 sec
✓	10	23:28:03	COMMIT	0 row(s) affected	0.031 sec

Time	Action	Message	Duration/Fetch
22:58:04	START TRANSACTION	0 row(s) affected	0.000 sec
22:58:04	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/data1.txt' INTO TABLE orders FIELDS TERMINATED BY ' ' LINES TERMINATED BY '\r\n'	5000000 row(s) affected Records: 5000000 Deleted: 0 Skipped: 0 Warnings: 0	94.844 sec
22:59:39	COMMIT	0 row(s) affected	8.578 sec
23:28:03	START TRANSACTION	0 row(s) affected	0.000 sec
23:28:03	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/data2.txt' INTO TABLE products FIELDS TERMINATED BY ' ' LINES TERMINATED BY '\r\n' IGNORE 0 LINES	10000 row(s) affected Records: 10000 Deleted: 0 Skipped: 0 Warnings: 0	0.625 sec
23:28:03	COMMIT	0 row(s) affected	0.031 sec

附：建表命令

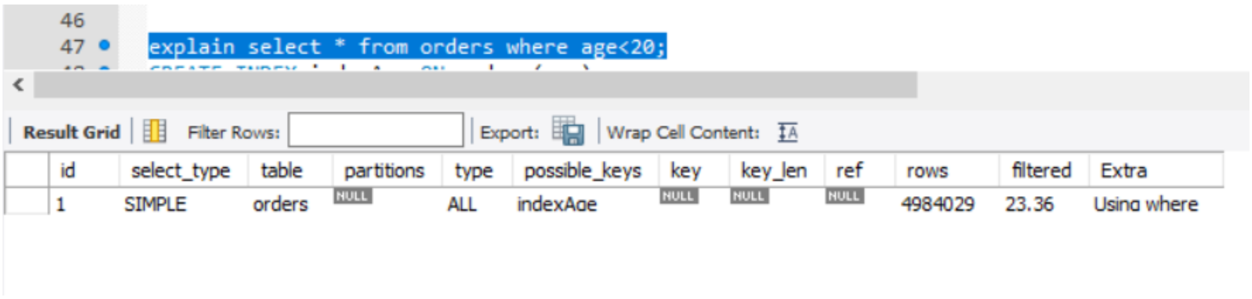


```sql
CREATE TABLE IF NOT EXISTS `orders` (
  `id` int unsigned NOT NULL auto_increment,
  `name` varchar(8) NOT NULL,
  `age` tinyint NOT NULL,
  `sex` enum('男','女'),
  `amount` double(12,2) NOT NULL,
  PRIMARY KEY (`id`)
)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
CREATE TABLE IF NOT EXISTS `products` (
  `id` smallint unsigned NOT NULL auto_increment,
  `pid` varchar(16) NOT NULL,
  `nums` smallint NOT NULL,
  PRIMARY KEY (`id`)
)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

3. 问题1：在 orders 表中找出购买人年龄小于20岁的order列表。

   SQL: select * from orders where age<20;

   不能建立索引理由: 使用age索引的话，就查询条件age<20而言，区分度不高，在查询的结果中，结果集的数据行占了表中数据行的很大比例，即需要在表中搜索的数据行的比例很大。增加索引，并不能明显加快检索速度，实际反而会更慢，如图sql优化器会直接选择全表扫描。



```
explain select * from orders where age<20;
```

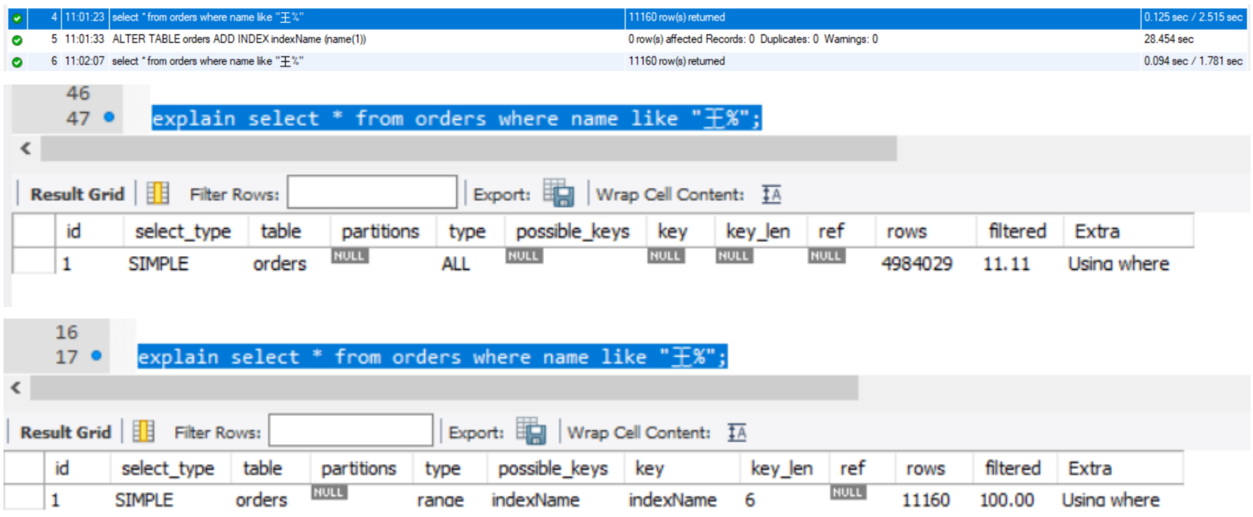
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	orders	NULL	ALL	indexAge	NULL	NULL	NULL	4984029	23.36	Using where

4. 问题2：在 orders 表中找出所有姓王的人的order列表。

   SQL: select * from orders where name like "王%";

   建立索引方式：ALTER TABLE orders ADD INDEX indexName (name(1));

   建立索引前后执行效率截图：



```
4  11:01:23  select * from orders where name like "王%"            11160 row(s) returned                                       0.125 sec / 2.515 sec
5  11:01:33  ALTER TABLE orders ADD INDEX indexName (name(1))    0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0     28.454 sec
6  11:02:07  select * from orders where name like "王%"            11160 row(s) returned                                       0.094 sec / 1.781 sec
```

```
explain select * from orders where name like "王%";
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	orders	NULL	ALL	NULL	NULL	NULL	NULL	4984029	11.11	Using where

```
explain select * from orders where name like "王%";
```

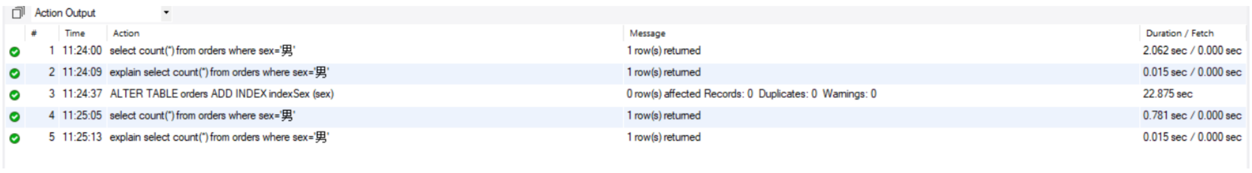
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	orders	NULL	range	indexName	indexName	6	NULL	11160	100.00	Using where

5. 问题3：统计 orders 表中所有男性的人的数量。

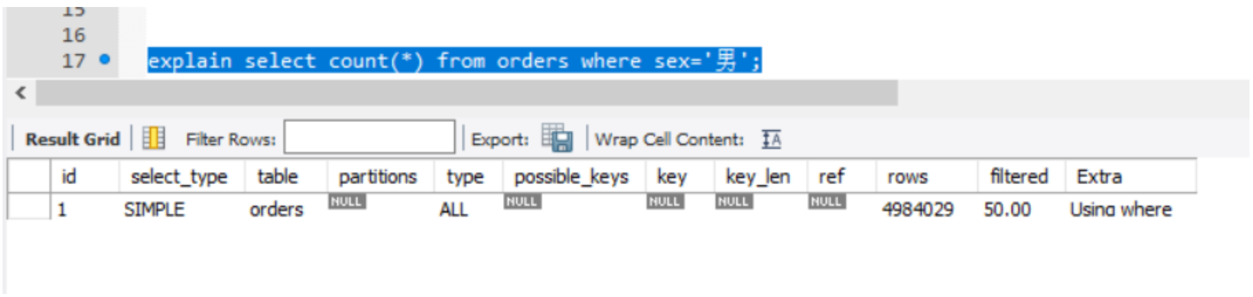
   SQL: select count(*) from orders where sex='男';

   建立索引方式：ALTER TABLE orders ADD INDEX indexSex (sex);

   建立索引前后执行效率截图：(只需统计列数而不用获取具体列内容，可建立indexSex 索引)



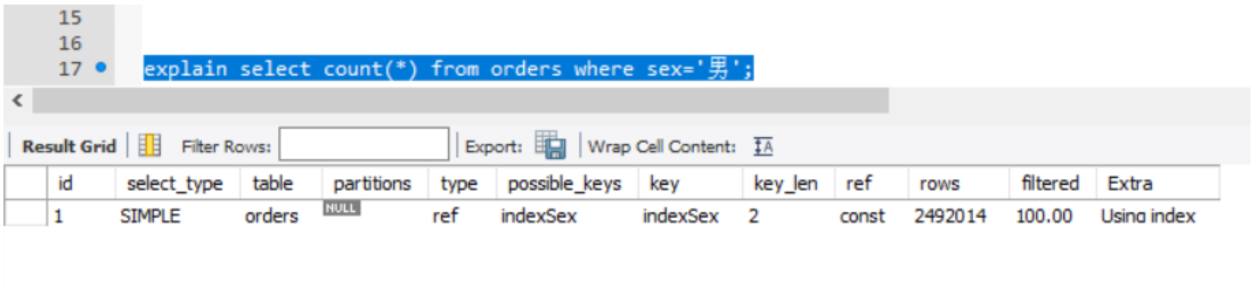
```
Action Output
#  Time      Action                                              Message                                                     Duration / Fetch
1  11:24:00  select count(*) from orders where sex='男'           1 row(s) returned                                           2.062 sec / 0.000 sec
2  11:24:09  explain select count(*) from orders where sex='男'   1 row(s) returned                                           0.015 sec / 0.000 sec
3  11:24:37  ALTER TABLE orders ADD INDEX indexSex (sex)         0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0    22.875 sec
4  11:25:05  select count(*) from orders where sex='男'           1 row(s) returned                                           0.781 sec / 0.000 sec
5  11:25:13  explain select count(*) from orders where sex='男'   1 row(s) returned                                           0.015 sec / 0.000 sec
```



```
explain select count(*) from orders where sex='男';
```

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

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	orders	NULL	ALL	NULL	NULL	NULL	NULL	4984029	50.00	Using where



```
explain select count(*) from orders where sex='男';
```

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

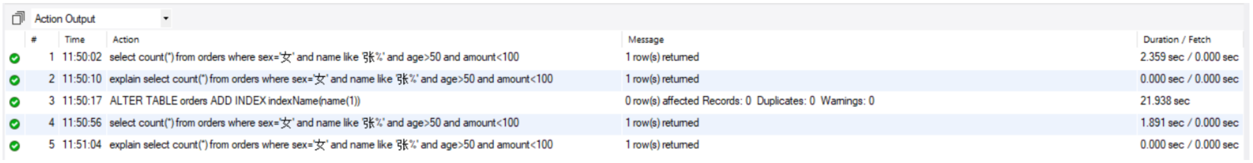
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	orders	NULL	ref	indexSex	indexSex	2	const	2492014	100.00	Using index

6. 问题4：在 orders 表中计算女性，姓张，年龄大于50，且消费小于100的人数。

SQL: select count(*) from orders where sex='女' and name like '张%' and age>50 and amount<100;

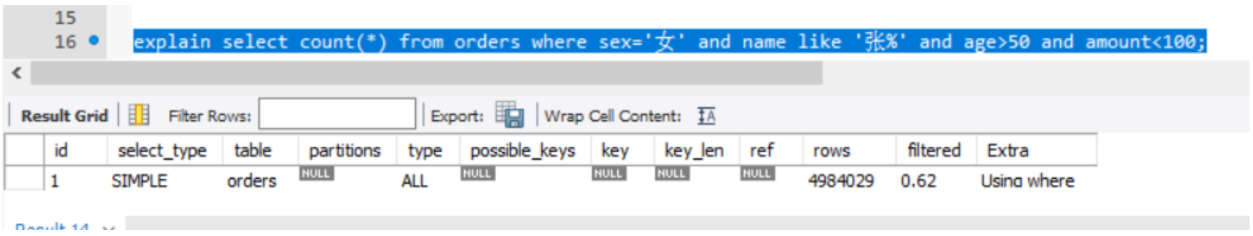
建立索引方式：ALTER TABLE orders ADD INDEX indexName(name(1));

建立索引前后执行效率截图：



Action Output

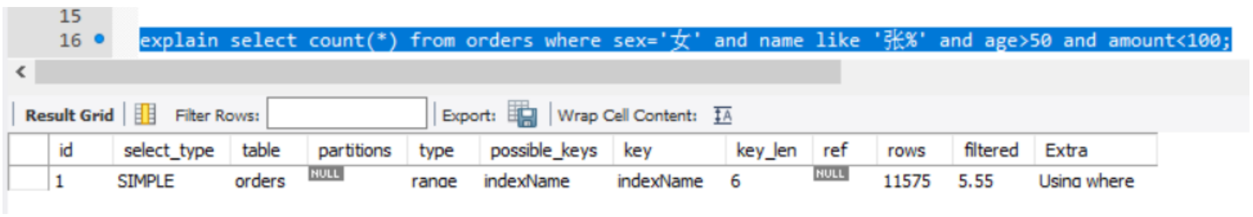
#	Time	Action	Message	Duration / Fetch
1	11:50:02	select count(*) from orders where sex='女' and name like '张%' and age>50 and amount<100	1 row(s) returned	2.359 sec / 0.000 sec
2	11:50:10	explain select count(*) from orders where sex='女' and name like '张%' and age>50 and amount<100	1 row(s) returned	0.000 sec / 0.000 sec
3	11:50:17	ALTER TABLE orders ADD INDEX indexName(name(1))	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	21.938 sec
4	11:50:56	select count(*) from orders where sex='女' and name like '张%' and age>50 and amount<100	1 row(s) returned	1.891 sec / 0.000 sec
5	11:51:04	explain select count(*) from orders where sex='女' and name like '张%' and age>50 and amount<100	1 row(s) returned	0.000 sec / 0.000 sec



```
explain select count(*) from orders where sex='女' and name like '张%' and age>50 and amount<100;
```

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

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	orders	NULL	ALL	NULL	NULL	NULL	NULL	4984029	0.62	Using where



```
explain select count(*) from orders where sex='女' and name like '张%' and age>50 and amount<100;
```

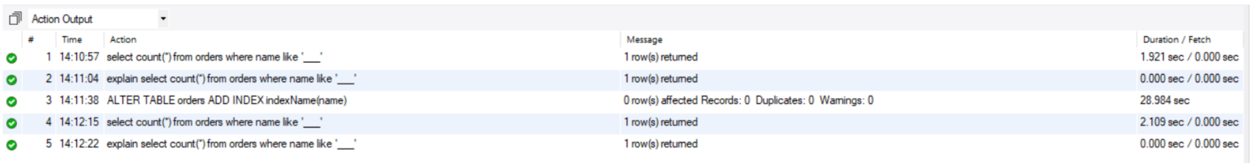
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	orders	NULL	range	indexName	indexName	6	NULL	11575	5.55	Using where

7. 问题5：统计 orders 表中姓名为三个字的人数。

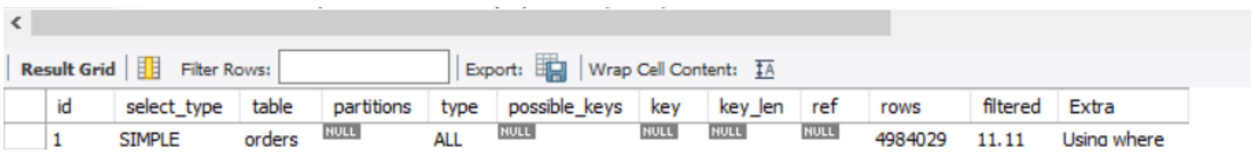
SQL: select count(*) from orders where name like '___';

不能建立索引理由: 即便在name列建立索引，就无前置定式的模糊查询条件like '___'而言，并不能明显加快检索速度，实际反而会更慢（见下图）。

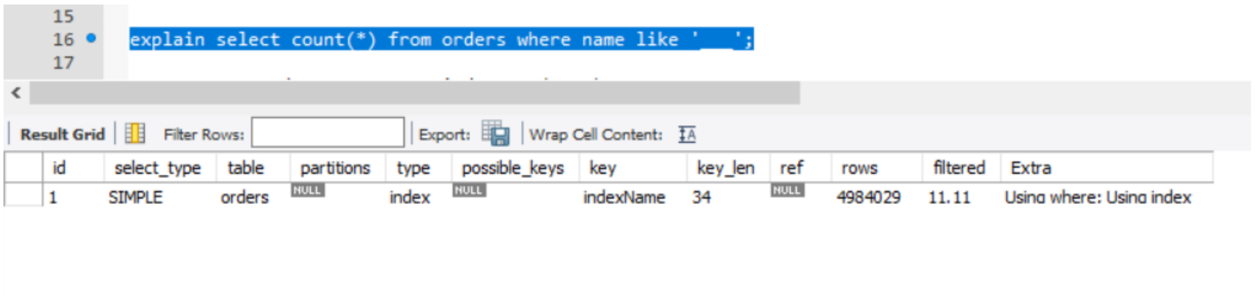


Action Output

#	Time	Action	Message	Duration / Fetch
1	14:10:57	select count(*) from orders where name like '___'	1 row(s) returned	1.921 sec / 0.000 sec
2	14:11:04	explain select count(*) from orders where name like '___'	1 row(s) returned	0.000 sec / 0.000 sec
3	14:11:38	ALTER TABLE orders ADD INDEX indexName(name)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	28.984 sec
4	14:12:15	select count(*) from orders where name like '___'	1 row(s) returned	2.109 sec / 0.000 sec
5	14:12:22	explain select count(*) from orders where name like '___'	1 row(s) returned	0.000 sec / 0.000 sec



id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	orders	NULL	ALL	NULL	NULL	NULL	NULL	4984029	11.11	Using where



```
explain select count(*) from orders where name like '___';
```

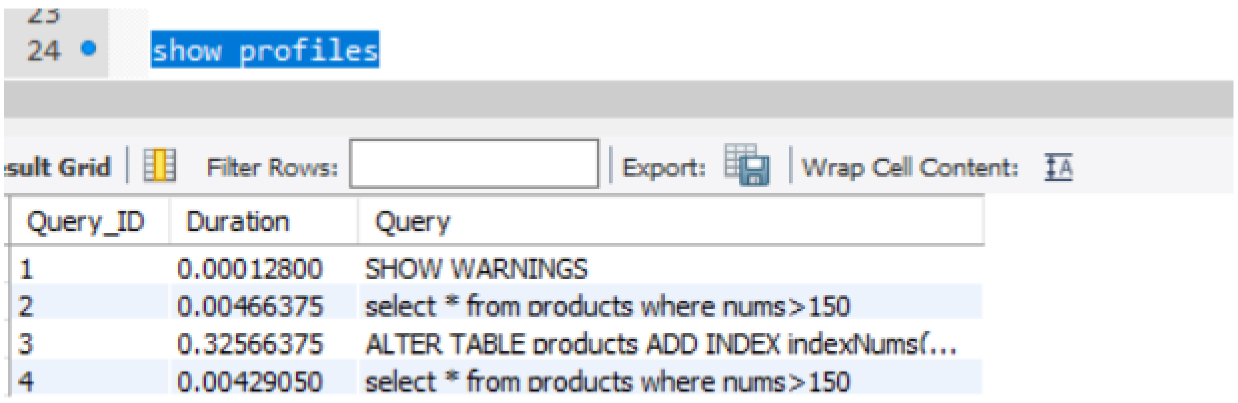
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	orders	NULL	index	NULL	indexName	34	NULL	4984029	11.11	Using where; Using index

8. 问题6：在 products 表中查找库存大于150的product列表。

   SQL: select * from products where nums>150;

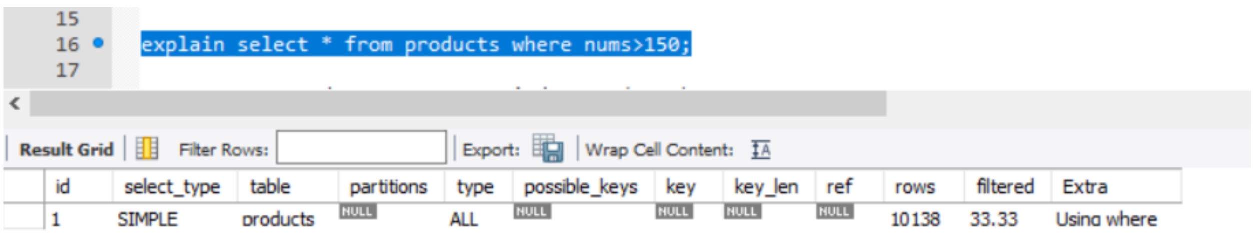
   建立索引方式：ALTER TABLE products ADD INDEX indexNums(nums);

   建立索引前后执行效率截图：



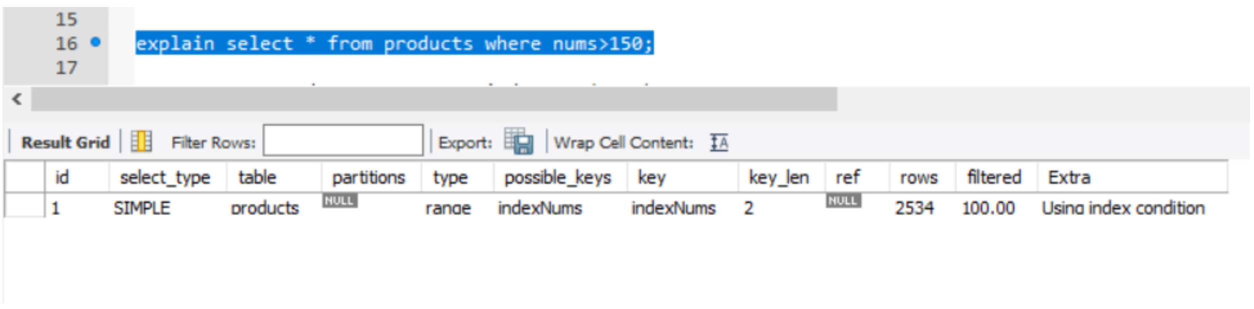
```
show profiles
```

Query_ID	Duration	Query
1	0.00012800	SHOW WARNINGS
2	0.00466375	select * from products where nums>150
3	0.32566375	ALTER TABLE products ADD INDEX indexNums(...
4	0.00429050	select * from products where nums>150



```
explain select * from products where nums>150;
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	products	NULL	ALL	NULL	NULL	NULL	NULL	10138	33.33	Using where



```
explain select * from products where nums>150;
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	products	NULL	range	indexNums	indexNums	2	NULL	2534	100.00	Using index condition