1. 插入 orders 表的插入方式操作描述，时间截图。

   load data local infile '/Users/cst/Desktop/数据库开发技术/作业/第一次作业/data1.txt' into table orders

```
mysql> load data local infile '/Users/cst/Desktop/数据库开发技术/作业/第一次作业
/data1.txt' into table orders;
Query OK, 5000000 rows affected (1 min 26.91 sec)
Records: 5000000  Deleted: 0  Skipped: 0  Warnings: 0
```

2. 插入 products 表的插入方式操作描述，时间截图。

   load data local infile '/Users/cst/Desktop/数据库开发技术/作业/第一次作业/data2.txt' into table products

```
mysql> load data local infile '/Users/cst/Desktop/数据库开发技术/作业/第一次作业
/data2.txt' into table products;
Query OK, 10000 rows affected (0.30 sec)
Records: 10000  Deleted: 0  Skipped: 0  Warnings: 0
```

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;

   建立索引方式：create index age_index on orders(age);

```
mysql> create index age_index on orders(age);
Query OK, 0 rows affected (7.79 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

   建立索引前后执行效率截图：

   前：

   查询时间：

```
571196 rows in set (1.84 sec)
```

   查询计划：

```
mysql> 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  | NULL          | NULL | NULL
 | NULL | 4992516 |    33.33 | Using where |
+----+-------------+--------+------------+------+---------------+------+---------
-+------+---------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

   后：

   查询时间：

```
571196 rows in set (1.87 sec)
```

查询计划：

```
mysql> 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  | age_index     | NULL | NULL
 | NULL | 4992516 |    23.36 | Using where |
+----+-------------+--------+------------+------+--------------+------+---------
-+------+---------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

结果：仍使用全表查询

解释：首先、age列上的不重复值较少(70个)，选择性低(70/5000000=0.000014)，索引查询效率低下。而且本次查询到的数据量较大、索引后匹配数据行的时间开销也很大，所以最终导致使用全表查询。

```
mysql> select count(distinct age) from orders;
+----------------------+
| count(distinct age) |
+----------------------+
|                   70 |
+----------------------+
1 row in set (0.00 sec)
```

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：create index name_index on orders(name);

```
mysql> create index name_index on orders(name);
Query OK, 0 rows affected (10.53 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

   建立索引前后执行效率截图：

   前：

   查询时间：

```
11160 rows in set (1.69 sec)
```

   查询计划：

```
mysql> 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 | 4992516 |    11.11 | Using where |
+----+-------------+--------+------------+------+---------------+------+---------
-+------+---------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

后：

查询时间：

```
11160 rows in set (0.54 sec)
```

查询计划：

```
mysql> 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 | name_index    | name_index |
183     | NULL | 20630 |   100.00 | Using index condition; Using MRR |
+----+-------------+--------+------------+-------+---------------+------------+-
--------+------+-------+----------+---------------------------------+
1 row in set, 1 warning (0.01 sec)
```

结果：使用索引查询，大幅减小查询时间

解释：name列选择性高，且本次查询到的数据量小

```
mysql> select count(distinct name) from orders;
+-----------------------+
| count(distinct name) |
+-----------------------+
|               3477290 |
+-----------------------+
1 row in set (1.99 sec)
```

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：create index sex_index on orders(sex);

```
mysql> create index sex_index on orders(sex);
Query OK, 0 rows affected (6.77 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

   建立索引前后执行效率截图：

前：

查询时间：

```
mysql> select count(*) from orders where sex='男';
+----------+
| count(*) |
+----------+
|  2499997 |
+----------+
1 row in set (1.22 sec)
```

查询计划：

```
mysql> explain select count(*) from orders where sex='男';
+----+-------------+--------+------------+------+---------------+------+---------+------+---------+----------+-------------+
| id | select_type | table  | partitions | type | possible_keys | key  | key_len | ref  | rows    | filtered | Extra       |
+----+-------------+--------+------------+------+---------------+------+---------+------+---------+----------+-------------+
|  1 | SIMPLE      | orders | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 4992516 |    50.00 | Using where |
+----+-------------+--------+------------+------+---------------+------+---------+------+---------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

后：

查询时间：

```
mysql> select count(*) from orders where sex='男';
+----------+
| count(*) |
+----------+
|  2499997 |
+----------+
1 row in set (0.43 sec)
```

查询计划：

```
mysql> explain select count(*) from orders where sex='男';
+----+-------------+--------+------------+------+---------------+-----------+---------+-------+---------+----------+-------------+
| id | select_type | table  | partitions | type | possible_keys | key       | key_len | ref   | rows    | filtered | Extra       |
+----+-------------+--------+------------+------+---------------+-----------+---------+-------+---------+----------+-------------+
|  1 | SIMPLE      | orders | NULL       | ref  | sex_index     | sex_index | 2       | const | 2496258 |   100.00 | Using index |
+----+-------------+--------+------------+------+---------------+-----------+---------+-------+---------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

结果：使用索引查询，减小查询时间

解释：虽然sex列选择性低（只有 '男'、'女'两种），但该查询只需统计索引个数，无须匹配数据行，所以使用该索引可以加快查询速度。（若是查询所有性别为男的 order 列表，该索引意义不大）

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：基于sex的全值匹配和其余列的选择性的高低

```
mysql> select count(distinct name), count(distinct amount), count( distinct age)
 from orders;
+----------------------+-----------------------+---------------------+
| count(distinct name) | count(distinct amount) | count( distinct age) |
+----------------------+-----------------------+---------------------+
|              3477290 |                100001 |                  70 |
+----------------------+-----------------------+---------------------+
1 row in set (14.48 sec)
```

   建立如下索引：

   create index create index four_index on orders(sex, name, amount, age);

```
mysql> create index four_index on orders(sex, name, amount, age);
Query OK, 0 rows affected (14.73 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

   建立索引前后执行效率截图：

   前：

   查询时间：

```
mysql> select count(*) from orders where sex = '女' and name like '张%' and age
> 50 and amount < 100;
+----------+
| count(*) |
+----------+
|      258 |
+----------+
1 row in set (1.89 sec)
```

   查询计划：

```
mysql> explain select count(*) from orders where sex = '女' and name like '张%'
and age > 50 and amount < 100;
+----+-------------+--------+------------+------+---------------+------+--------
-+------+---------+----------+-------------+
| id | select_type | table  | partitions | type | possible_keys | key  | key_len
 | ref  | rows    | filtered | Extra       |
+----+-------------+--------+------------+------+---------------+------+--------
-+------+---------+----------+-------------+
|  1 | SIMPLE      | orders | NULL       | ALL  | NULL          | NULL | NULL
 | NULL | 4992516 |     0.62 | Using where |
+----+-------------+--------+------------+------+---------------+------+--------
-+------+---------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

   后：

   查询时间：

```
mysql> select count(*) from orders where sex = '女' and name like '张%' and age
> 50 and amount < 100;
+----------+
| count(*) |
+----------+
|      258 |
+----------+
1 row in set (0.01 sec)
```

查询计划：

```
mysql> explain select count(*) from orders where sex = '女' and name like '张%'
and age > 50 and amount < 100;
+----+-------------+--------+------------+-------+---------------+------------+-
--------+------+------+----------+-------------------------+
| id | select_type | table  | partitions | type  | possible_keys | key        |
key_len | ref  | rows | filtered | Extra                   |
+----+-------------+--------+------------+-------+---------------+------------+-
--------+------+------+----------+-------------------------+
|  1 | SIMPLE      | orders | NULL       | range | four_index    | four_index |
189     | NULL | 5715 |    11.11 | Using where; Using index |
+----+-------------+--------+------------+-------+---------------+------------+-
--------+------+------+----------+-------------------------+
1 row in set, 1 warning (0.00 sec)
```

结果：使用复合索引，大幅减小查询时间

解释：该索引为复合索引（因为name是范围索引，所以只使用了索引的sex和name列），选择性高且只需查询索引项即可

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 '___';

   无法使用了索引

   理由：以%或_开头的查询无法使用索引（不符合列前缀的要求）

   建立索引前后执行效率截图：

   前：

   查询时间：

```
mysql> select count(*) from orders where name like '___';
+----------+
| count(*) |
+----------+
|  2501252 |
+----------+
1 row in set (1.36 sec)
```

   查询计划：

```
mysql> 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       | ALL  | NULL          | NULL | NULL
 | NULL | 4987097 |    11.11 | Using where |
+----+-------------+--------+------------+------+---------------+------+---------
-+------+---------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

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：create index nums_index on products(nums);

```
mysql> create index nums_index on products(nums);
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

建立索引前后执行效率截图：

前：

查询时间：

```
2534 rows in set (0.01 sec)
```

查询计划：

```
mysql> 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 | 9963 |    33.33 | Using where |
+----+-------------+----------+------------+------+---------------+------+---------+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

后：

查询时间：

```
2534 rows in set (0.01 sec)
```

查询计划：

```
mysql> 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 | nums_index    | nums_index | 5       | NULL | 2534 |   100.00 | Using index condition |
+----+-------------+----------+------------+-------+---------------+-------------+---------+------+------+----------+-----------------------+
1 row in set, 1 warning (0.00 sec)
```

结论：使用了索引，但数据量较小，效果不明显