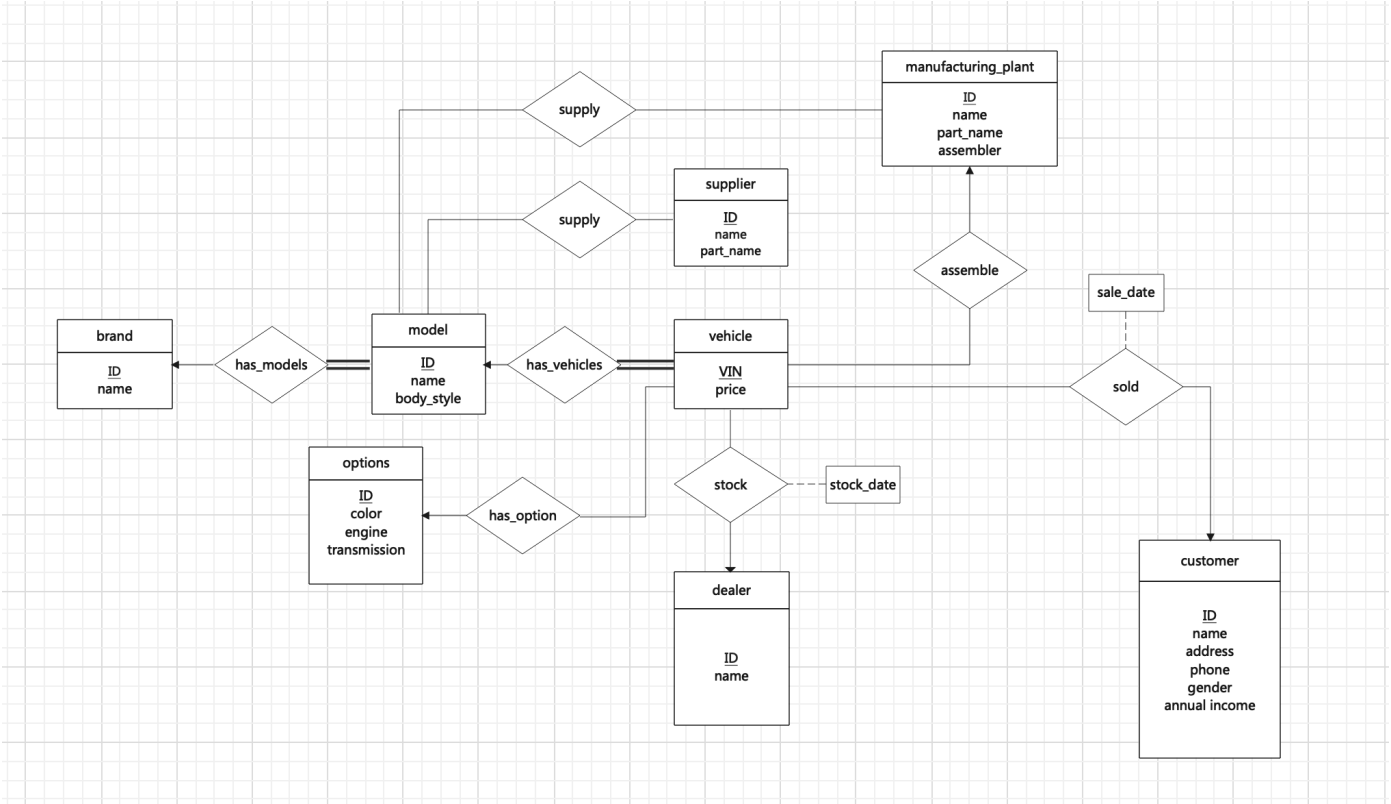
Database Course Project

I.E-R Model



- 这里我们假定每一辆vehicle只由一家manufacturing plant来组装；
- 每一个model都能够得到不同supplier提供的零部件；
- stock联系集表示dealer从manufacturer进货，stock_date表示进货的日期，dealer可以选择售出他所拥有的vehicle，出售日期由sold联系集上的sale_date属性决定，如果vehicle的VID没有出现在sold联系集中，则表示这辆vehicle没有被dealer出售；
- vehicle和model实体集分别在has_vehicles和has_models联系集中全部参与（用双实线表示）；

II.Relation Model

vehicle (VIN, model_id, option_id, plant_id, dealer_id, price, stock_date)

brand (brand_id, brand_name)

model (model_id, brand_id, model_name, body_style)

options(option_id, color, engine, transmission)

dealer (dealer_id, dealer_name)

customer (customer_id, customer_name, customer_address, phone, gender, annual_income)

supplier (supplier_id, supplier_name, part_name)

manufacturing_plant (plant_id, plant_name, part_name, assembler)

supply (supplier_id, model_id)

sold(VIN, customer_id, sale_date)

判断以上关系模式是否符合BC范式（Boyce-Codd Normal Form，BCNF），即：

对于 F^+ 中所有形如 $\alpha \rightarrow \beta$ 的函数依赖（其中 $\alpha \subseteq R$，$\beta \subseteq R$），下面至少有一项成立：

$$\alpha \rightarrow \beta \text{是平凡函数依赖}$$
$$\alpha \text{是模式} R \text{的一个超码}$$

在vehicle关系模式中，主键为VIN，存在非平凡函数依赖：VIN $\rightarrow$ model_id, option_id, plant_id, dealer_id, price, stock_date，满足BC范式的要求；

在sold关系模式中，主键为VIN和customer_id，存在非平凡函数依赖: VIN, customer_id $\rightarrow$ sale_date，满足BC范式的要求；

类似地，经过验证，以上所有关系模式均符合BC范式的要求；

III.Queries

- Show **sales trends** for various brands over the past 3 years, by year, month, week. Then break these data out by gender of the buyer and then by income range.

这里，对于sale trend，我们针对每个brand的销量进行查询，并查看其趋势：

首先，按照每个brand在前三年中，每年的销量进行查询：

```
select brand_name, count(brand_id) as unit_sales, year(sale_date) as year
  from vehicle natural join sold natural join model natural join brand
  where sale_date between '2021-01-01' and '2023-12-31'
  group by year, brand_name
  order by brand_name, year desc;
```

```
mysql> select brand_name, count(brand_id) as unit_sales, year(sale_date) as year
    ->   from vehicle natural join sold natural join model natural join brand
    ->   where sale_date between '2021-01-01' and '2023-12-31'
    ->   group by year, brand_name
    ->  order by brand_name, year desc;
+------------+------------+------+
| brand_name | unit_sales | year |
+------------+------------+------+
| Audi       |          1 | 2022 |
| Audi       |          2 | 2021 |
| BMW        |          3 | 2022 |
| FORD       |          3 | 2022 |
| Honda      |          2 | 2023 |
| Honda      |          1 | 2022 |
| Jaguar     |          4 | 2023 |
| Kia        |          2 | 2023 |
| Kia        |          2 | 2022 |
| Mazda      |          3 | 2023 |
| Peugeot    |          3 | 2022 |
| Porsche    |          1 | 2022 |
| Porsche    |          2 | 2021 |
| Tesla      |          1 | 2023 |
| Tesla      |          2 | 2021 |
+------------+------------+------+
15 rows in set (0.01 sec)

mysql>
```

Audi在2022～2021年间，分别出售了1和2台，在2021年销量最大；

之后，再按照月份查看其趋势：

```sql
select brand_name, count(brand_id) as unit_sales, year(sale_date) as year,
month(sale_date) as month
  from vehicle natural join sold natural join model natural join brand
  where sale_date between '2021-01-01' and '2023-12-31'
  group by year, brand_name, month
  order by brand_name, year desc, month desc;
```

```
mysql> select brand_name, count(brand_id) as unit_sales, year(sale_date) as year, month(sale_date) as month
    ->   from vehicle natural join sold natural join model natural join brand
    ->   where sale_date between '2021-01-01' and '2023-12-31'
    ->  group by year, brand_name, month
    -> order by brand_name, year desc, month desc;
+------------+------------+------+-------+
| brand_name | unit_sales | year | month |
+------------+------------+------+-------+
| Audi       |          1 | 2022 |     4 |
| Audi       |          1 | 2021 |     9 |
| Audi       |          1 | 2021 |     4 |
| BMW        |          1 | 2022 |    12 |
| BMW        |          1 | 2022 |    11 |
| BMW        |          1 | 2022 |    10 |
| FORD       |          1 | 2022 |     5 |
| FORD       |          1 | 2022 |     2 |
| FORD       |          1 | 2022 |     1 |
| Honda      |          2 | 2023 |     1 |
| Honda      |          1 | 2022 |    11 |
| Jaguar     |          4 | 2023 |     1 |
| Kia        |          2 | 2023 |     1 |
| Kia        |          1 | 2022 |     4 |
| Kia        |          1 | 2022 |     3 |
| Mazda      |          3 | 2023 |     1 |
| Peugeot    |          1 | 2022 |    11 |
| Peugeot    |          1 | 2022 |     7 |
| Peugeot    |          1 | 2022 |     2 |
| Porsche    |          1 | 2022 |     4 |
| Porsche    |          2 | 2021 |     1 |
| Tesla      |          1 | 2023 |     1 |
| Tesla      |          2 | 2021 |     1 |
+------------+------------+------+-------+
23 rows in set (0.00 sec)

mysql>
```

这里将每年每月的销量都展示出来，一些月份没有产生交易的在这里没有记录，因此没有显示；

```
select brand_name, count(brand_id) as unit_sales, year(sale_date) as year,
month(sale_date) as month, week(sale_date) as week
  from vehicle natural join sold natural join model natural join brand
  where sale_date between '2021-01-01' and '2023-12-31'
  group by year, brand_name, month, week
  order by brand_name, year desc, month;
```

```
mysql> select brand_name, count(brand_id) as unit_sales, year(sale_date) as year, month(sale_date) as month, week(sale_date) as week
    ->   from vehicle natural join sold natural join model natural join brand
    ->   where sale_date between '2021-01-01' and '2023-12-31'
    ->  group by year, brand_name, month, week
    -> order by brand_name, year desc, month;
+------------+------------+------+-------+------+
| brand_name | unit_sales | year | month | week |
+------------+------------+------+-------+------+
| Audi       |          1 | 2022 |     4 |   13 |
| Audi       |          1 | 2021 |     4 |   17 |
| Audi       |          1 | 2021 |     9 |   37 |
| BMW        |          1 | 2022 |    10 |   41 |
| BMW        |          1 | 2022 |    11 |   47 |
| BMW        |          1 | 2022 |    12 |   51 |
| FORD       |          1 | 2022 |     1 |    1 |
| FORD       |          1 | 2022 |     2 |    7 |
| FORD       |          1 | 2022 |     5 |   20 |
| Honda      |          2 | 2023 |     1 |    2 |
| Honda      |          1 | 2022 |    11 |   44 |
| Jaguar     |          4 | 2023 |     1 |    1 |
| Kia        |          2 | 2023 |     1 |    1 |
| Kia        |          1 | 2022 |     3 |    9 |
| Kia        |          1 | 2022 |     4 |   14 |
| Mazda      |          3 | 2023 |     1 |    1 |
| Peugeot    |          1 | 2022 |     2 |    5 |
| Peugeot    |          1 | 2022 |     7 |   28 |
| Peugeot    |          1 | 2022 |    11 |   47 |
| Porsche    |          1 | 2022 |     4 |   17 |
| Porsche    |          2 | 2021 |     1 |    2 |
| Tesla      |          1 | 2023 |     1 |    1 |
| Tesla      |          1 | 2021 |     1 |    1 |
| Tesla      |          1 | 2021 |     1 |    0 |
+------------+------------+------+-------+------+
24 rows in set (0.00 sec)

mysql>
```

以上就是每个brand每年每月每周的销售额的查询结果，如此可以从销售额的角度，分析出每个brand的sale trend；

最后，将这些数据按照customer的gender和income range进行拆分：

```
select customer_id, gender, annual_income, brand_id, brand_name, count(brand_id) as
unit_sales, year(sale_date) as year, month(sale_date) as month, week(sale_date) as week
  from vehicle natural join sold natural join model natural join brand natural join
customer
  where sale_date between '2021-01-01' and '2023-12-31'
  group by year, brand_name, month, week, brand_id, customer_id, gender, annual_income
  order by brand_name, year desc, month;
```

```
mysql> select customer_id, gender, annual_income, brand_id, brand_name, count(brand_id) as unit_sales, year(sale_date) as year, month(sale_date) as month, week(sale_date) as week
    ->   from vehicle natural join sold natural join model natural join brand natural join customer
    ->   where sale_date between '2021-01-01' and '2023-12-31'
    ->  group by year, brand_name, month, week, brand_id, customer_id, gender, annual_income
    -> order by brand_name, year desc, month;
+-------------+--------+---------------+----------+------------+------------+------+-------+------+
| customer_id | gender | annual_income | brand_id | brand_name | unit_sales | year | month | week |
+-------------+--------+---------------+----------+------------+------------+------+-------+------+
| 19991       | F      |      90000.00 | 10000    | Audi       |          1 | 2022 |     4 |   13 |
| 12345       | F      |      90000.00 | 10000    | Audi       |          1 | 2021 |     4 |   17 |
| 44553       | F      |     120000.00 | 10000    | Audi       |          1 | 2021 |     9 |   37 |
| 45678       | F      |      90000.00 | 10001    | BMW        |          1 | 2022 |    10 |   41 |
| 54321       | M      |      90000.00 | 10001    | BMW        |          1 | 2022 |    11 |   47 |
| 70557       | F      |      70000.00 | 10001    | BMW        |          1 | 2022 |    12 |   51 |
| 98765       | F      |     190000.00 | 10002    | FORD       |          1 | 2022 |     1 |    1 |
| 76653       | M      |      40000.00 | 10002    | FORD       |          1 | 2022 |     2 |    7 |
| 98988       | M      |      20000.00 | 10002    | FORD       |          1 | 2022 |     5 |   20 |
| 70557       | F      |      70000.00 | 10007    | Honda      |          1 | 2023 |     1 |    2 |
| 76653       | M      |      40000.00 | 10007    | Honda      |          1 | 2023 |     1 |    2 |
| 12345       | F      |      90000.00 | 10007    | Honda      |          1 | 2022 |    11 |   44 |
| 98988       | M      |      20000.00 | 10008    | Jaguar     |          1 | 2023 |     1 |    1 |
| 70557       | F      |      70000.00 | 10008    | Jaguar     |          1 | 2023 |     1 |    1 |
| 76543       | F      |     200000.00 | 10008    | Jaguar     |          1 | 2023 |     1 |    1 |
| 54321       | M      |      90000.00 | 10008    | Jaguar     |          1 | 2023 |     1 |    1 |
| 76653       | M      |      40000.00 | 10005    | Kia        |          1 | 2023 |     1 |    1 |
| 12345       | F      |      90000.00 | 10005    | Kia        |          1 | 2023 |     1 |    1 |
| 98988       | M      |      20000.00 | 10005    | Kia        |          1 | 2022 |     3 |    9 |
| 98988       | M      |      20000.00 | 10005    | Kia        |          1 | 2022 |     4 |   14 |
| 70557       | F      |      70000.00 | 10009    | Mazda      |          1 | 2023 |     1 |    1 |
| 54321       | M      |      90000.00 | 10009    | Mazda      |          1 | 2023 |     1 |    1 |
| 98988       | M      |      20000.00 | 10009    | Mazda      |          1 | 2023 |     1 |    1 |
| 12345       | F      |      90000.00 | 10006    | Peugeot    |          1 | 2022 |     2 |    5 |
| 12345       | F      |      90000.00 | 10006    | Peugeot    |          1 | 2022 |     7 |   28 |
| 76653       | M      |      40000.00 | 10006    | Peugeot    |          1 | 2022 |    11 |   47 |
| 23121       | M      |      90000.00 | 10003    | Porsche    |          1 | 2022 |     4 |   17 |
| 55739       | F      |      90000.00 | 10003    | Porsche    |          2 | 2021 |     1 |    2 |
| 70557       | F      |      70000.00 | 10004    | Tesla      |          1 | 2023 |     1 |    1 |
| 55739       | F      |      90000.00 | 10004    | Tesla      |          1 | 2021 |     1 |    0 |
| 70557       | F      |      70000.00 | 10004    | Tesla      |          1 | 2021 |     1 |    1 |
+-------------+--------+---------------+----------+------------+------------+------+-------+------+
31 rows in set (0.01 sec)

mysql>
```

这里显示了购买这些brand的customer的gender和annual_income；

之后按照gender进行划分：

> 男性

```
select customer_id, gender, annual_income, brand_id, brand_name, count(brand_id) as
unit_sales, year(sale_date) as year, month(sale_date) as month, week(sale_date) as week
  from vehicle natural join sold natural join model natural join brand natural join
customer
  where sale_date between '2021-01-01' and '2023-12-31' and gender = 'M'
  group by year, brand_name, month, week, brand_id, customer_id, gender, annual_income
  order by brand_name, year, month desc;
```

```
mysql> select customer_id, gender, annual_income, brand_id, brand_name, count(brand_id) as unit_sales, year(sale_date) as year, month(sale_date) as month, week(sale_date) as week
    ->   from vehicle natural join sold natural join model natural join brand natural join customer
    ->   where sale_date between '2021-01-01' and '2023-12-31' and gender = 'M'
    ->  group by year, brand_name, month, week, brand_id, customer_id, gender, annual_income
    -> order by brand_name, year, month desc;
+-------------+--------+---------------+----------+------------+------------+------+-------+------+
| customer_id | gender | annual_income | brand_id | brand_name | unit_sales | year | month | week |
+-------------+--------+---------------+----------+------------+------------+------+-------+------+
| 54321       | M      |      90000.00 | 10001    | BMW        |          1 | 2022 |    11 |   47 |
| 98988       | M      |      20000.00 | 10002    | FORD       |          1 | 2022 |     5 |   20 |
| 76653       | M      |      40000.00 | 10002    | FORD       |          1 | 2022 |     2 |    7 |
| 76653       | M      |      40000.00 | 10007    | Honda      |          1 | 2023 |     1 |    2 |
| 54321       | M      |      90000.00 | 10008    | Jaguar     |          1 | 2023 |     1 |    1 |
| 98988       | M      |      20000.00 | 10008    | Jaguar     |          1 | 2023 |     1 |    1 |
| 98988       | M      |      20000.00 | 10005    | Kia        |          1 | 2022 |     4 |   14 |
| 98988       | M      |      20000.00 | 10005    | Kia        |          1 | 2022 |     3 |    9 |
| 76653       | M      |      40000.00 | 10005    | Kia        |          1 | 2023 |     1 |    1 |
| 54321       | M      |      90000.00 | 10009    | Mazda      |          1 | 2023 |     1 |    1 |
| 98988       | M      |      20000.00 | 10009    | Mazda      |          1 | 2023 |     1 |    1 |
| 76653       | M      |      40000.00 | 10006    | Peugeot    |          1 | 2022 |    11 |   47 |
| 23121       | M      |      90000.00 | 10003    | Porsche    |          1 | 2022 |     4 |   17 |
+-------------+--------+---------------+----------+------------+------------+------+-------+------+
13 rows in set (0.00 sec)
```

> 女性

```
select customer_id, gender, annual_income, brand_id, brand_name, count(brand_id) as
unit_sales, year(sale_date) as year, month(sale_date) as month, week(sale_date) as week
  from vehicle natural join sold natural join model natural join brand natural join
customer
  where sale_date between '2021-01-01' and '2023-12-31' and gender = 'F'
  group by year, brand_name, month, week, brand_id, customer_id, gender, annual_income
  order by brand_name, year, month desc;
```

```
mysql> select customer_id, gender, annual_income, brand_id, brand_name, count(brand_id) as unit_sales, year(sale_date) as year, month(sale_date) as month, week(sale_date) as week
    ->   from vehicle natural join sold natural join model natural join brand natural join customer
    ->   where sale_date between '2021-01-01' and '2023-12-31' and gender = 'F'
    ->  group by year, brand_name, month, week, brand_id, customer_id, gender, annual_income
    -> order by brand_name, year, month desc;
+-------------+--------+---------------+----------+------------+------------+------+-------+------+
| customer_id | gender | annual_income | brand_id | brand_name | unit_sales | year | month | week |
+-------------+--------+---------------+----------+------------+------------+------+-------+------+
| 44553       | F      |     120000.00 | 10000    | Audi       |          1 | 2021 |     9 |   37 |
| 12345       | F      |      90000.00 | 10000    | Audi       |          1 | 2021 |     4 |   17 |
| 19991       | F      |      90000.00 | 10000    | Audi       |          1 | 2022 |     4 |   13 |
| 70557       | F      |      70000.00 | 10001    | BMW        |          1 | 2022 |    12 |   51 |
| 45678       | F      |      90000.00 | 10001    | BMW        |          1 | 2022 |    10 |   41 |
| 98765       | F      |     190000.00 | 10002    | FORD       |          1 | 2022 |     1 |    1 |
| 12345       | F      |      90000.00 | 10007    | Honda      |          1 | 2022 |    11 |   44 |
| 70557       | F      |      70000.00 | 10007    | Honda      |          1 | 2023 |     1 |    2 |
| 70557       | F      |      70000.00 | 10008    | Jaguar     |          1 | 2023 |     1 |    1 |
| 76543       | F      |     200000.00 | 10008    | Jaguar     |          1 | 2023 |     1 |    1 |
| 12345       | F      |      90000.00 | 10005    | Kia        |          1 | 2023 |     1 |    1 |
| 70557       | F      |      70000.00 | 10009    | Mazda      |          1 | 2023 |     1 |    1 |
| 12345       | F      |      90000.00 | 10006    | Peugeot    |          1 | 2022 |     7 |   28 |
| 12345       | F      |      90000.00 | 10006    | Peugeot    |          1 | 2022 |     2 |    5 |
| 55739       | F      |      90000.00 | 10003    | Porsche    |          2 | 2021 |     1 |    2 |
| 55739       | F      |      90000.00 | 10004    | Tesla      |          1 | 2021 |     1 |    0 |
| 70557       | F      |      70000.00 | 10004    | Tesla      |          1 | 2021 |     1 |    1 |
| 70557       | F      |      70000.00 | 10004    | Tesla      |          1 | 2023 |     1 |    1 |
+-------------+--------+---------------+----------+------------+------------+------+-------+------+
18 rows in set (0.01 sec)
```

最后按照annual_income的range进行划分：

```
select customer_id, gender, annual_income, brand_id, brand_name, count(brand_id) as
unit_sales, year(sale_date) as year, month(sale_date) as month, week(sale_date) as week
  from vehicle natural join sold natural join model natural join brand natural join
customer
  where sale_date between '2021-01-01' and '2023-12-31' and annual_income > 70000
  group by year, brand_name, month, week, brand_id, customer_id, gender, annual_income
  order by brand_name, year, month desc;
```

对于annual_income > 70000 的customer:

```
mysql>   select customer_id, gender, annual_income, brand_id, brand_name, count(brand_id) as unit_sales, year(sale_date) as year, month(sale_date) as month, week(sale_date) as week
    ->   from vehicle natural join sold natural join model natural join brand natural join customer
    ->   where sale_date between '2021-01-01' and '2023-12-31' and annual_income > 70000
    ->  group by year, brand_name, month, week, brand_id, customer_id, gender, annual_income
    -> order by brand_name, year, month desc;
+-------------+--------+---------------+----------+------------+------------+------+-------+------+
| customer_id | gender | annual_income | brand_id | brand_name | unit_sales | year | month | week |
+-------------+--------+---------------+----------+------------+------------+------+-------+------+
| 44553       | F      |     120000.00 | 10000    | Audi       |          1 | 2021 |     9 |   37 |
| 12345       | F      |      90000.00 | 10000    | Audi       |          1 | 2021 |     4 |   17 |
| 19991       | F      |      90000.00 | 10000    | Audi       |          1 | 2022 |     4 |   13 |
| 54321       | M      |      90000.00 | 10001    | BMW        |          1 | 2022 |    11 |   47 |
| 45678       | F      |      90000.00 | 10001    | BMW        |          1 | 2022 |    10 |   41 |
| 98765       | F      |     190000.00 | 10002    | FORD       |          1 | 2022 |     1 |    1 |
| 12345       | F      |      90000.00 | 10007    | Honda      |          1 | 2022 |    11 |   44 |
| 54321       | M      |      90000.00 | 10008    | Jaguar     |          1 | 2023 |     1 |    1 |
| 76543       | F      |     200000.00 | 10008    | Jaguar     |          1 | 2023 |     1 |    1 |
| 12345       | F      |      90000.00 | 10005    | Kia        |          1 | 2023 |     1 |    1 |
| 54321       | M      |      90000.00 | 10009    | Mazda      |          1 | 2023 |     1 |    1 |
| 12345       | F      |      90000.00 | 10006    | Peugeot    |          1 | 2022 |     7 |   28 |
| 12345       | F      |      90000.00 | 10006    | Peugeot    |          1 | 2022 |     2 |    5 |
| 55739       | F      |      90000.00 | 10003    | Porsche    |          2 | 2021 |     1 |    2 |
| 23121       | M      |      90000.00 | 10003    | Porsche    |          1 | 2022 |     4 |   17 |
| 55739       | F      |      90000.00 | 10004    | Tesla      |          1 | 2021 |     1 |    0 |
+-------------+--------+---------------+----------+------------+------------+------+-------+------+
16 rows in set (0.09 sec)

mysql>
```

```
select customer_id, gender, annual_income, brand_id, brand_name, count(brand_id) as
unit_sales, year(sale_date) as year, month(sale_date) as month, week(sale_date) as week
  from vehicle natural join sold natural join model natural join brand natural join
customer
  where sale_date between '2021-01-01' and '2023-12-31' and annual_income <= 70000
  group by year, brand_name, month, week, brand_id, customer_id, gender, annual_income
  order by brand_name, year, month desc;
```

对于annual_income <= 70000 的customer:

```
mysql> select customer_id, gender, annual_income, brand_id, brand_name, count(brand_id) as unit_sales, year(sale_date) as year, month(sale_date) as month, week(sale_date) as week
    ->   from vehicle natural join sold natural join model natural join brand natural join customer
    ->   where sale_date between '2021-01-01' and '2023-12-31' and annual_income <= 70000
    ->  group by year, brand_name, month, week, brand_id, customer_id, gender, annual_income
    -> order by brand_name, year, month desc;
+-------------+--------+---------------+----------+------------+------------+------+-------+------+
| customer_id | gender | annual_income | brand_id | brand_name | unit_sales | year | month | week |
+-------------+--------+---------------+----------+------------+------------+------+-------+------+
| 70557       | F      |      70000.00 | 10001    | BMW        |          1 | 2022 |    12 |   51 |
| 98988       | M      |      20000.00 | 10002    | FORD       |          1 | 2022 |     5 |   20 |
| 76653       | M      |      40000.00 | 10002    | FORD       |          1 | 2022 |     2 |    7 |
| 70557       | F      |      70000.00 | 10007    | Honda      |          1 | 2023 |     1 |    2 |
| 76653       | M      |      40000.00 | 10007    | Honda      |          1 | 2023 |     1 |    2 |
| 70557       | F      |      70000.00 | 10008    | Jaguar     |          1 | 2023 |     1 |    1 |
| 98988       | M      |      20000.00 | 10008    | Jaguar     |          1 | 2023 |     1 |    1 |
| 98988       | M      |      20000.00 | 10005    | Kia        |          1 | 2022 |     4 |   14 |
| 98988       | M      |      20000.00 | 10005    | Kia        |          1 | 2022 |     3 |    9 |
| 76653       | M      |      40000.00 | 10005    | Kia        |          1 | 2023 |     1 |    1 |
| 70557       | F      |      70000.00 | 10009    | Mazda      |          1 | 2023 |     1 |    1 |
| 98988       | M      |      20000.00 | 10009    | Mazda      |          1 | 2023 |     1 |    1 |
| 76653       | M      |      40000.00 | 10006    | Peugeot    |          1 | 2022 |    11 |   47 |
| 70557       | F      |      70000.00 | 10004    | Tesla      |          1 | 2021 |     1 |    1 |
| 70557       | F      |      70000.00 | 10004    | Tesla      |          1 | 2023 |     1 |    1 |
+-------------+--------+---------------+----------+------------+------------+------+-------+------+
15 rows in set (0.01 sec)

mysql>
```

- Suppose that it is found that **transmissions** made by **supplier Getrag** between **two given dates** are defective. Find the **VIN** of each car containing such a transmission and the **customer** to which it was sold. If your design allows, suppose the defective transmissions all come from only one of Getrag's plants.

这里设定在2021-01-01到2022-12-30期间，供应商Getrag所提供的transmission是损坏的；

```
create view vehicle_supplier_sold(VID, supplier_name, customer_id, sale_date) as
  select VIN, supplier_name, customer_id, sale_date
  from vehicle natural join model natural join supplier natural join sold;

select VID, supplier_name, customer_id, sale_date
  from vehicle_supplier_sold
  where supplier_name = 'Getrag' and sale_date between '2021-01-01' and '2022-12-30';
```

```
mysql> select VID, supplier_name, customer_id, sale_date
    -> from vehicle_supplier_sold
    -> where supplier_name = 'Getrag' and sale_date between '2021-01-01' and '2022-12-30';
+-------------------+---------------+-------------+------------+
| VID               | supplier_name | customer_id | sale_date  |
+-------------------+---------------+-------------+------------+
| 1FVXA7AS24LM58815 | Getrag        | 76653       | 2022-02-15 |
| 1G1ZT51806F128009 | Getrag        | 76653       | 2022-11-25 |
| 1GCCS1956Y8235348 | Getrag        | 98765       | 2022-01-05 |
| 1GCEK14K8RE106083 | Getrag        | 12345       | 2022-07-15 |
| 1GNDT13S582215117 | Getrag        | 98988       | 2022-03-05 |
| 1HD1GPM15CC339172 | Getrag        | 12345       | 2022-02-05 |
| 2B3HD46R02H210893 | Getrag        | 12345       | 2022-11-05 |
| 5N3ZA0NE6AN906847 | Getrag        | 54321       | 2022-11-23 |
| JH4DA1850JS005062 | Getrag        | 70557       | 2022-12-24 |
| JH4DA3350GS005185 | Getrag        | 98988       | 2022-05-15 |
| JH4DC2380SS000012 | Getrag        | 98988       | 2022-04-05 |
| JTDBE30K620061417 | Getrag        | 23121       | 2022-04-25 |
| WBAAM3333XFP59732 | Getrag        | 45678       | 2022-10-11 |
+-------------------+---------------+-------------+------------+
13 rows in set (0.06 sec)
```

如此可以查询到在2021-01-01到2022-12-30期间使用供应商Getrag所提供的transmission而制成的vehicle出售的customer的信息；

若要进一步查询customer的更多信息，可以执行如下代码：

```sql
select VID, supplier_name, customer_id, sale_date, customer_name, customer_address,
phone, gender, annual_income
  from vehicle_supplier_sold natural join customer
  where supplier_name = 'Getrag' and sale_date between '2021-01-01' and '2022-12-30';
```

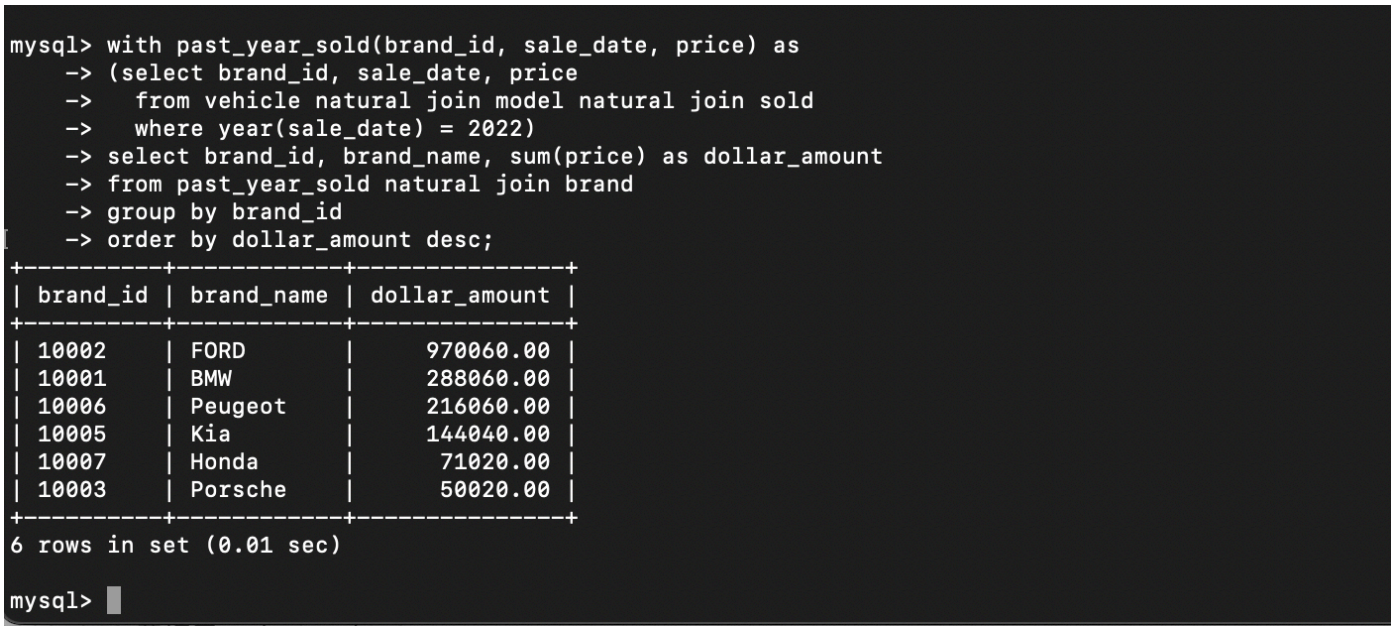
```
mysql> select VID, supplier_name, customer_id, sale_date, customer_name, customer_address, phone, gender, annual_income
    ->   from vehicle_supplier_sold natural join customer
    ->   where supplier_name = 'Getrag' and sale_date between '2021-01-01' and '2022-12-30';
+-------------------+---------------+-------------+------------+---------------+------------------+---------+--------+---------------+
| VID               | supplier_name | customer_id | sale_date  | customer_name | customer_address | phone   | gender | annual_income |
+-------------------+---------------+-------------+------------+---------------+------------------+---------+--------+---------------+
| 1GCEK14K8RE106083 | Getrag        | 12345       | 2022-07-15 | Shankar       | Nottingham       | 3210000 | F      |      90000.00 |
| 1HD1GPM15CC339172 | Getrag        | 12345       | 2022-02-05 | Shankar       | Nottingham       | 3210000 | F      |      90000.00 |
| 2B3HD46R02H210893 | Getrag        | 12345       | 2022-11-05 | Shankar       | Nottingham       | 3210000 | F      |      90000.00 |
| JTDBE30K620061417 | Getrag        | 23121       | 2022-04-25 | Chavez        | Toronto          | 1110001 | M      |      90000.00 |
| WBAAM3333XFP59732 | Getrag        | 45678       | 2022-10-11 | Levy          | Montreal         | 4610001 | F      |      90000.00 |
| 5N3ZA0NE6AN906847 | Getrag        | 54321       | 2022-11-23 | Williams      | Havana           | 5410009 | M      |      90000.00 |
| JH4DA1850JS005062 | Getrag        | 70557       | 2022-12-24 | Snow          | Chicago          | 0100122 | F      |      70000.00 |
| 1FVXA7AS24LM58815 | Getrag        | 76653       | 2022-02-15 | Aoi           | Havana           | 6010003 | M      |      40000.00 |
| 1G1ZT51806F128009 | Getrag        | 76653       | 2022-11-25 | Aoi           | Havana           | 6010003 | M      |      40000.00 |
| 1GCCS1956Y8235348 | Getrag        | 98765       | 2022-01-05 | Bourikas      | Nottingham       | 9810010 | F      |     190000.00 |
| 1GNDT13S582215117 | Getrag        | 98988       | 2022-03-05 | Tanaka        | Beijing          | 1201000 | M      |      20000.00 |
| JH4DA3350GS005185 | Getrag        | 98988       | 2022-05-15 | Tanaka        | Beijing          | 1201000 | M      |      20000.00 |
| JH4DC2380SS000012 | Getrag        | 98988       | 2022-04-05 | Tanaka        | Beijing          | 1201000 | M      |      20000.00 |
+-------------------+---------------+-------------+------------+---------------+------------------+---------+--------+---------------+
13 rows in set (0.03 sec)

mysql>
```

这样就补充了customer的更多信息：name, address, phone, gender, annual_income；

- Find the top 2 brands by dollar-amount sold in the past year.

Past year：我们这里认为是2022年

```
with past_year_sold(brand_id, sale_date, price) as
(select brand_id, sale_date, price
  from vehicle natural join model natural join sold
  where year(sale_date) = 2022)
select brand_id, brand_name, sum(price) as dollar_amount
  from past_year_sold natural join brand
  group by brand_id
  order by dollar_amount desc
  limit 2;
```



```
mysql> with past_year_sold(brand_id, sale_date, price) as
    -> (select brand_id, sale_date, price
    ->   from vehicle natural join model natural join sold
    ->   where year(sale_date) = 2022)
    -> select brand_id, brand_name, sum(price) as dollar_amount
    -> from past_year_sold natural join brand
    -> group by brand_id
    -> order by dollar_amount desc;
+----------+------------+---------------+
| brand_id | brand_name | dollar_amount |
+----------+------------+---------------+
| 10002    | FORD       |     970060.00 |
| 10001    | BMW        |     288060.00 |
| 10006    | Peugeot    |     216060.00 |
| 10005    | Kia        |     144040.00 |
| 10007    | Honda      |      71020.00 |
| 10003    | Porsche    |      50020.00 |
+----------+------------+---------------+
6 rows in set (0.01 sec)

mysql>
```

在不使用limit 2之前，查询的结果是按照dollar-amount从大到小排序的2022年（past year）每个brand所出售的情况；

使用limit 2后结果中显示的是top 2（按照dollar-amount排序最大和第二大）的brand的信息：

```
mysql> with past_year_sold(brand_id, sale_date, price) as
    -> (select brand_id, sale_date, price
    ->   from vehicle natural join model natural join sold
    ->   where year(sale_date) = 2022)
    -> select brand_id, brand_name, sum(price) as dollar_amount
    -> from past_year_sold natural join brand
    -> group by brand_id
    -> order by dollar_amount desc
    -> limit 2;
+----------+------------+---------------+
| brand_id | brand_name | dollar_amount |
+----------+------------+---------------+
| 10002    | FORD       |     970060.00 |
| 10001    | BMW        |     288060.00 |
+----------+------------+---------------+
2 rows in set (0.02 sec)

mysql>
```

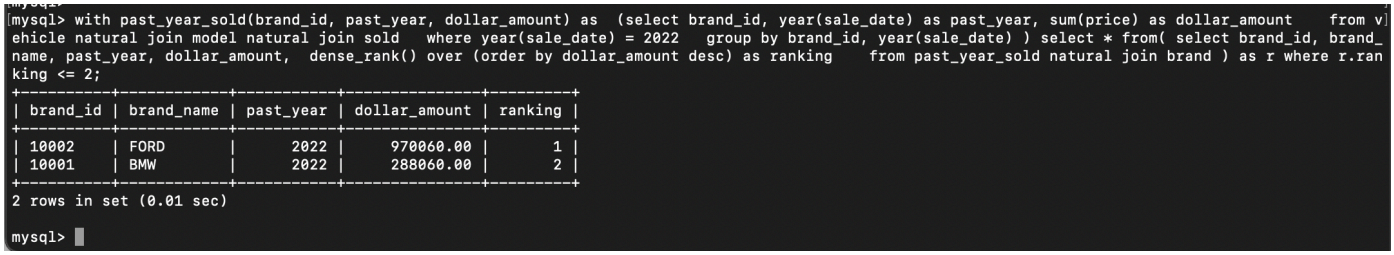
- 使用OLAP进行排序：

```
with past_year_sold(brand_id, past_year, dollar_amount) as
(select brand_id, year(sale_date) as past_year, sum(price) as dollar_amount
  from vehicle natural join model natural join sold
  where year(sale_date) = 2022
  group by brand_id, year(sale_date)
)
select brand_id, brand_name, past_year, dollar_amount,
       dense_rank() over (order by dollar_amount desc) ranking
       from past_year_sold natural join brand;
```

```
mysql> with past_year_sold(brand_id, past_year, dollar_amount) as  (select brand_id, year(sale_date) as past_year, sum(price) as dollar_amount    from v
ehicle natural join model natural join sold   where year(sale_date) = 2022   group by brand_id, year(sale_date) ) select brand_id, brand_name, past_year
, dollar_amount,  dense_rank() over (order by dollar_amount desc) ranking    from past_year_sold natural join brand;
+----------+------------+-----------+---------------+---------+
| brand_id | brand_name | past_year | dollar_amount | ranking |
+----------+------------+-----------+---------------+---------+
| 10002    | FORD       |      2022 |     970060.00 |       1 |
| 10001    | BMW        |      2022 |     288060.00 |       2 |
| 10006    | Peugeot    |      2022 |     216060.00 |       3 |
| 10005    | Kia        |      2022 |     144040.00 |       4 |
| 10000    | Audi       |      2022 |      80020.00 |       5 |
| 10007    | Honda      |      2022 |      71020.00 |       6 |
| 10003    | Porsche    |      2022 |      50020.00 |       7 |
+----------+------------+-----------+---------------+---------+
7 rows in set (0.01 sec)

mysql>
```

这里使用OLAP中的dense_rank进行排序，显示了去年（2022）每个brand的销售总额

```
with past_year_sold(brand_id, past_year, dollar_amount) as
(select brand_id, year(sale_date) as past_year, sum(price) as dollar_amount
  from vehicle natural join model natural join sold
  where year(sale_date) = 2022
  group by brand_id, year(sale_date)
)
select * from(
select brand_id, brand_name, past_year, dollar_amount,
       dense_rank() over (order by dollar_amount desc) as ranking
       from past_year_sold natural join brand
) as r
where r.ranking <= 2;
```



这样就能查询到top2销售额的brand;

- Find the top 2 brands by unit sales in the past year.

与上一个query差不多，只不过这里将总额的排序换成了销量的排序

```
with past_year_sold(brand_id, sale_date) as
(select brand_id, sale_date
  from vehicle natural join model natural join sold
  where year(sale_date) = 2022)
select brand_id, brand_name, count(sale_date) as unit_sales
  from past_year_sold natural join brand
  group by brand_id
  order by unit_sales desc
  limit 2;
```

不使用limit 2之前，查询的结果是按照2022年（past year）每个brand的销量从大到小排序的的情况；

```
mysql> with past_year_sold(brand_id, sale_date) as
    -> (select brand_id, sale_date
    ->   from vehicle natural join model natural join sold
    ->   where year(sale_date) = 2022)
    -> select brand_id, brand_name, count(sale_date) as unit_sales
    -> from past_year_sold natural join brand
    -> group by brand_id
    -> order by unit_sales desc;
+----------+------------+------------+
| brand_id | brand_name | unit_sales |
+----------+------------+------------+
| 10001    | BMW        |          3 |
| 10002    | FORD       |          3 |
| 10006    | Peugeot    |          3 |
| 10005    | Kia        |          2 |
| 10003    | Porsche    |          1 |
| 10007    | Honda      |          1 |
+----------+------------+------------+
6 rows in set (0.04 sec)

mysql>
```

使用limit 2后结果中显示的是top 2（按照销量排序最大和第二大）的brand的信息：

```
mysql> with past_year_sold(brand_id, sale_date) as
    -> (select brand_id, sale_date
    ->   from vehicle natural join model natural join sold
    ->   where year(sale_date) = 2022)
    -> select brand_id, brand_name, count(sale_date) as unit_sales
    -> from past_year_sold natural join brand
    -> group by brand_id
    -> order by unit_sales desc
    -> limit 2;
+----------+------------+------------+
| brand_id | brand_name | unit_sales |
+----------+------------+------------+
| 10001    | BMW        |          3 |
| 10002    | FORD       |          3 |
+----------+------------+------------+
2 rows in set (0.01 sec)

mysql>
```

- 使用OLAP进行排序：

```
with past_year_sold(brand_id, past_year, unit_sales) as
(select brand_id, year(sale_date) as past_year, count(price) as unit_sales
  from vehicle natural join model natural join sold
  where year(sale_date) = 2022
  group by brand_id, year(sale_date)
)
select * from(
select brand_id, brand_name, past_year, unit_sales,
       dense_rank() over (order by unit_sales desc) as ranking
       from past_year_sold natural join brand
) as r
where r.ranking <= 2;
```

```
+----------+------------+-----------+------------+---------+
| brand_id | brand_name | past_year | unit_sales | ranking |
+----------+------------+-----------+------------+---------+
| 10002    | FORD       |      2022 |          3 |       1 |
| 10006    | Peugeot    |      2022 |          3 |       1 |
| 10001    | BMW        |      2022 |          3 |       1 |
| 10005    | Kia        |      2022 |          2 |       2 |
+----------+------------+-----------+------------+---------+
4 rows in set (0.01 sec)

mysql>
```

以上是OLAP得到的排序结果，有三个排名第一的brand，Kia排名第二；

- In what month(s) do **convertibles** sell best?

```
select body_style, month(sale_date) as month, count(model_id) as sell_count
  from vehicle natural join model natural join sold
  where body_style = 'Convertible'
  group by month
  order by sell_count desc;
```

```
mysql> select body_style, month(sale_date) as month, count(model_id) as sell_count
    ->   from vehicle natural join model natural join sold
    ->   where body_style = 'Convertible'
    ->   group by month
    ->   order by sell_count desc;
+-------------+-------+------------+
| body_style  | month | sell_count |
+-------------+-------+------------+
| Convertible |     1 |          3 |
| Convertible |     4 |          1 |
| Convertible |     5 |          1 |
| Convertible |    11 |          1 |
+-------------+-------+------------+
4 rows in set (0.00 sec)

mysql>
```

从以上查询结果可知，Convertible在1月的销量最好，达到3台；

- Find those dealers who keep a vehicle in inventory for the longest average time.

这里需要查询的是将vehicle作为库存时间最长的dealer，换句话说，是求那些直至今日都还没有出售vehicle或者已经出售，与购买这辆vehicle的时间之差最大：

```
select dealer_id, dealer_name, datediff(sale_date, stock_date) as inventory_time
  from vehicle natural join sold natural join dealer
  order by inventory_time desc;
```

```
mysql>   select dealer_id, dealer_name, datediff(sale_date, stock_date) as inventory_time
    ->   from vehicle natural join sold natural join dealer
    ->   order by inventory_time desc;
+-----------+-------------+----------------+
| dealer_id | dealer_name | inventory_time |
+-----------+-------------+----------------+
| 98345     | Kim         |           4589 |
| 10101     | Srinivasan  |           4365 |
| 10101     | Srinivasan  |           4079 |
| 45565     | Katz        |           4061 |
| 10101     | Srinivasan  |           3992 |
| 22222     | Einstein    |           3926 |
| 15151     | Mozart      |           3900 |
| 98345     | Kim         |           3736 |
| 10101     | Srinivasan  |           3705 |
| 10101     | Srinivasan  |           3524 |
| 22222     | Einstein    |           3381 |
| 10101     | Srinivasan  |           1606 |
| 33456     | Gold        |           1443 |
| 10101     | Srinivasan  |           1233 |
| 12121     | Wu          |           1146 |
| 98345     | Kim         |           1092 |
| 98345     | Kim         |           1067 |
| 32343     | El Said     |           1047 |
| 98345     | Kim         |            997 |
| 10101     | Srinivasan  |            906 |
| 76766     | Crick       |            834 |
| 33456     | Gold        |            785 |
| 76766     | Crick       |            764 |
| 22222     | Einstein    |            762 |
| 45565     | Katz        |            740 |
| 98345     | Kim         |            728 |
| 45565     | Katz        |            725 |
| 45565     | Katz        |            644 |
| 10101     | Srinivasan  |            635 |
| 10101     | Srinivasan  |            470 |
| 15151     | Mozart      |            360 |
| 15151     | Mozart      |            346 |
+-----------+-------------+----------------+
32 rows in set (0.01 sec)

mysql>
```

从结果可以看出，Kim是将vehicle作为库存时间最长的dealer，达到了4589天；

IV. Interfaces

- The database administrator (you) may use SQL either via the command line or SQL Developer.
- The marketing department needs sales reports and may want to do special data mining and analysis. Provide them with a simple interface to generate some OLAP results.

这里administrator的interface在终端连接SQL后输入相关的command即可， marketing department也可以通过command line的方式（如同在Query中一样）输入OLAP的查询语句；

- The vehicle locator service needs a lookup application to check inventory both locally and at nearby dealers.
- Online customers need an elegant Web interface to find dealers and check products, inventories, and prices.

Customer可以通过如下的Web interface来查询到每个dealer现有的vehicle的信息，如果vehicle没有被售出，那么在sold date字段显示的是Inventory，如果已经被售出，那么显示的是对应的售出日期，同时， vehicle locator service也可以通过以下的查询就可以知道对应的vehicle是否为inventory：

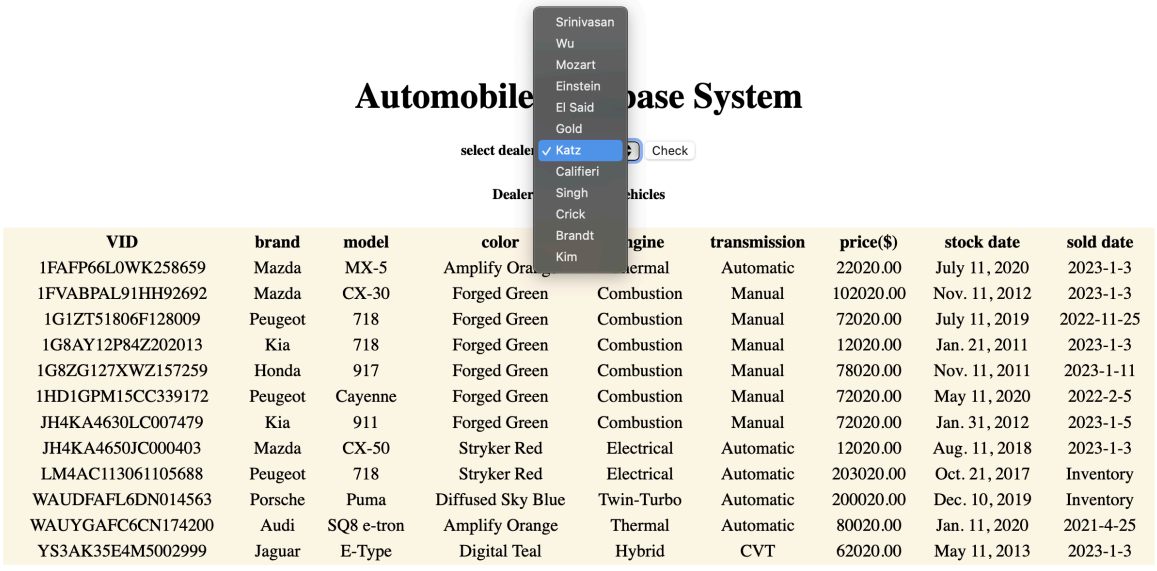
Automobile Database System

select dealer: Srinivasan Check

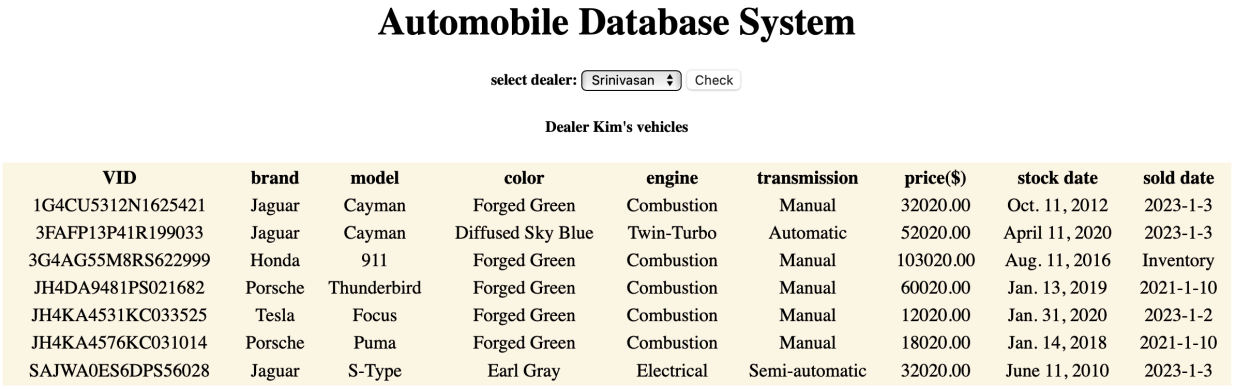
Dealer Srinivasan's vehicles

VID	brand	model	color	engine	transmission	price($)	stock date	sold date
1FAFP66L0WK258659	Mazda	MX-5	Amplify Orange	Thermal	Automatic	22020.00	July 11, 2020	2023-1-3
1FVABPAL91HH92692	Mazda	CX-30	Forged Green	Combustion	Manual	102020.00	Nov. 11, 2012	2023-1-3
1G1ZT51806F128009	Peugeot	718	Forged Green	Combustion	Manual	72020.00	July 11, 2019	2022-11-25
1G8AY12P84Z202013	Kia	718	Forged Green	Combustion	Manual	12020.00	Jan. 21, 2011	2023-1-3
1G8ZG127XWZ157259	Honda	917	Forged Green	Combustion	Manual	78020.00	Nov. 11, 2011	2023-1-11
1HD1GPM15CC339172	Peugeot	Cayenne	Forged Green	Combustion	Manual	72020.00	May 11, 2020	2022-2-5
JH4KA4630LC007479	Kia	911	Forged Green	Combustion	Manual	72020.00	Jan. 31, 2012	2023-1-5
JH4KA4650JC000403	Mazda	CX-50	Stryker Red	Electrical	Automatic	12020.00	Aug. 11, 2018	2023-1-3
LM4AC113061105688	Peugeot	718	Stryker Red	Electrical	Automatic	203020.00	Oct. 21, 2017	Inventory
WAUDFAFL6DN014563	Porsche	Puma	Diffused Sky Blue	Twin-Turbo	Automatic	200020.00	Dec. 10, 2019	Inventory
WAUYGAFC6CN174200	Audi	SQ8 e-tron	Amplify Orange	Thermal	Automatic	80020.00	Jan. 11, 2020	2021-4-25
YS3AK35E4M5002999	Jaguar	E-Type	Digital Teal	Hybrid	CVT	62020.00	May 11, 2013	2023-1-3

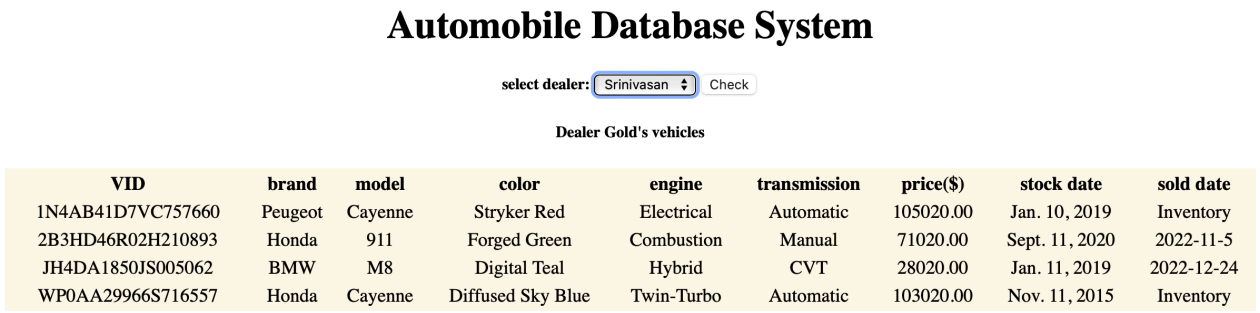
上图对应的是dealer Srinivasan拥有的vehicle的信息；



上图显示了目前有哪些dealer，我们这里认为dealer不重名；



上图对应的是dealer Kim的查询结果；



上图对应的是dealer Gold的查询结果；

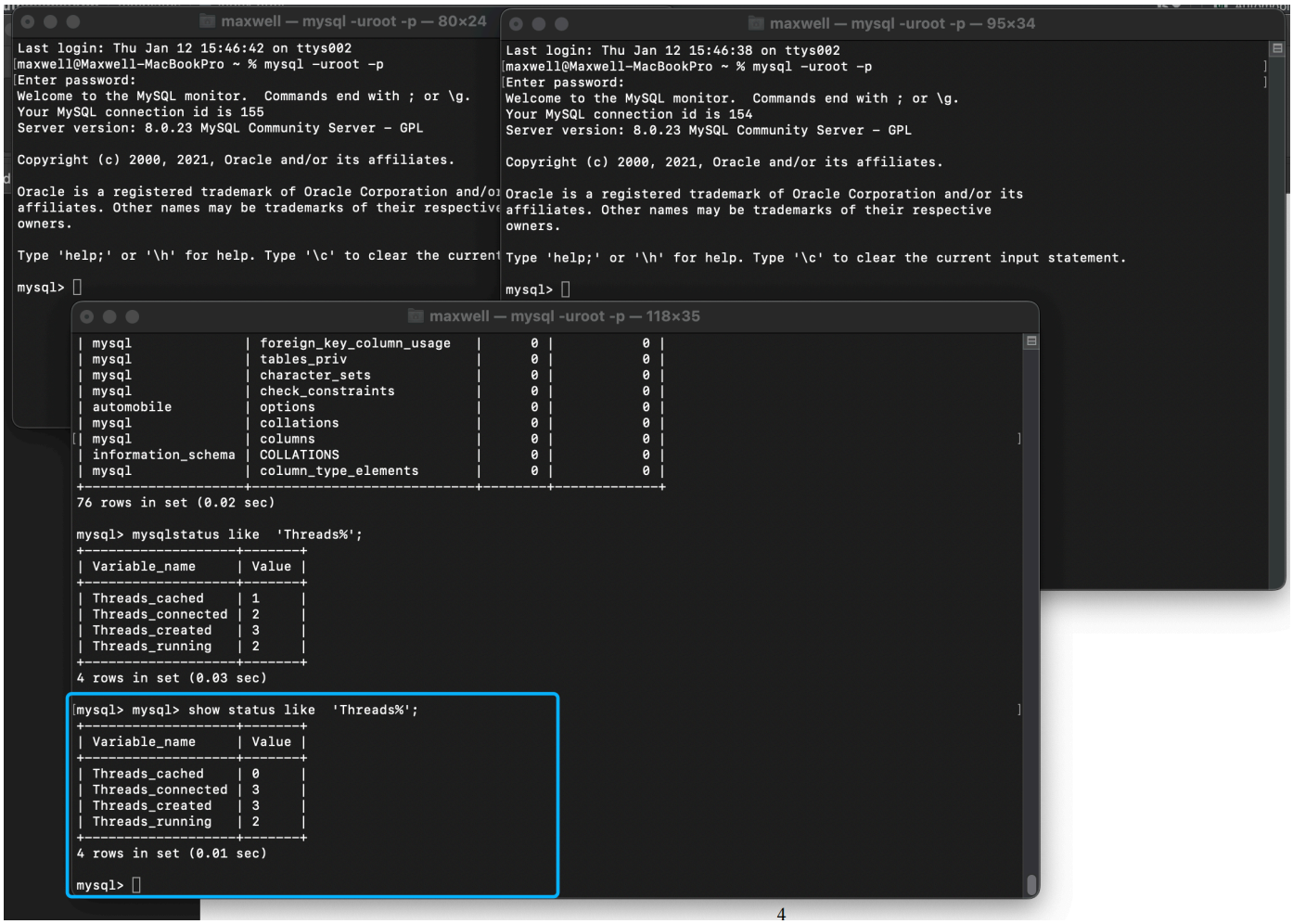
V. Concurrency

并发控制的目标是支持写入多于一个用户在同一时间完成交易的记录和新生成的车的记录；

mySQL支持多用户并发，但是并发时需要遵守一定的协议，对事务进行隔离，避免造成并发问题：脏读、不可重复读、幻读和丢失更新：

1. 脏读：事务 A 读取了事务 B 更新的数据，然后 B 回滚操作，那么 A 读取到的数据是脏数据
2. 不可重复读：事务 A 多次读取同一数据，事务 B 在事务 A 多次读取的过程中，对数据作了更新并提交，导致事务 A 多次读取同一数据时，结果不一致。
3. 幻读：系统管理员 A 将数据库中所有学生的成绩从具体分数改为 ABCDE 等级，但是系统管理员 B 就在这个时候插入了一条具体分数的记录，当系统管理员 A 改结束后发现还有一条记录没有改过来，就好像发生了幻觉一样，这就叫幻读。
4. 撤销一个事务时，把其他事务已提交的更新数据覆盖（A和B事务并发执行，A事务执行更新后，提交；B事务在A事务更新后，B事务结束前也做了对该行数据的更新操作，然后回滚，则两次更新操作都丢失了）。

为了测试并发的效果，这里开启三个终端连接同一个SQL服务器：



图中显示已连接的线程数（Threads_connected）为3

```
set autocommit = 0;
```

在vehicle表中，有这些记录，接下来以这个表作为并发控制的测试表：

```
mysql> select * from vehicle;
+-------------------+----------+-----------+----------+-----------+-----------+-------------+
| VIN               | model_id | option_id | plant_id | dealer_id | price     | stock_date  |
+-------------------+----------+-----------+----------+-----------+-----------+-------------+
| 1B4GP25R51B138763 | 20019    | 30000     | 80004    | 22222     | 303020.00 | 2018-12-18  |
| 1FAFP66L0WK258659 | 20027    | 30000     | 80003    | 10101     |  22020.00 | 2020-07-11  |
| 1FVABPAL91HH92692 | 20029    | 30002     | 80005    | 10101     | 102020.00 | 2012-11-11  |
| 1FVHCYDJ85HV14123 | 20023    | 30002     | 80007    | 22222     |  22020.00 | 2020-12-11  |
| 1FVXA7AS24LM58815 | 20006    | 30005     | 80006    | 45565     |  30020.00 | 2020-02-21  |
| 1G1ZT51806F128009 | 20020    | 30002     | 80000    | 10101     |  72020.00 | 2019-07-11  |
| 1G4CU5312N1625421 | 20024    | 30002     | 80006    | 98345     |  32020.00 | 2012-10-11  |
| 1G8AY12P84Z202013 | 20015    | 30002     | 80006    | 10101     |  12020.00 | 2011-01-21  |
| 1G8MF35X68Y131819 | 20012    | 30002     | 80003    | 15151     |  28020.00 | 2020-01-15  |
| 1G8ZG127XWZ157259 | 20022    | 30002     | 80002    | 10101     |  78020.00 | 2011-11-11  |
| 1GCCS1956Y8235348 | 20007    | 30002     | 80007    | 45565     | 900020.00 | 2020-04-01  |
| 1GCEK14K8RE106083 | 20019    | 30002     | 80002    | 76766     |  72020.00 | 2020-06-11  |
| 1GNDT13S582215117 | 20017    | 30002     | 80003    | 45565     |  72020.00 | 2011-01-21  |
| 1HD1GPM15CC339172 | 20018    | 30002     | 80001    | 10101     |  72020.00 | 2020-05-11  |
| 1N4AB41D7VC757660 | 20018    | 30001     | 80005    | 33456     | 105020.00 | 2019-01-10  |
| 1P3XP48K6LN273071 | 20001    | 30000     | 80001    | 12121     |  80020.00 | 2019-02-11  |
| 2B3HD46R02H210893 | 20021    | 30002     | 80001    | 33456     |  71020.00 | 2020-09-11  |
| 3FAFP13P41R199033 | 20024    | 30003     | 80002    | 98345     |  52020.00 | 2020-04-11  |
| 3G4AG55M8RS622999 | 20021    | 30002     | 80002    | 98345     | 103020.00 | 2016-08-11  |
| 5N3ZA0NE6AN906847 | 20004    | 30003     | 80004    | 32343     | 180020.00 | 2020-01-11  |
| JH4DA1850JS005062 | 20005    | 30004     | 80005    | 33456     |  28020.00 | 2019-01-11  |
| JH4DA3350GS005185 | 20008    | 30000     | 80008    | 45565     |  40020.00 | 2020-05-05  |
| JH4DA9481PS021682 | 20010    | 30002     | 80010    | 98345     |  60020.00 | 2019-01-13  |
| JH4DC2380SS000012 | 20016    | 30002     | 80004    | 22222     |  72020.00 | 2013-01-01  |
| JH4KA4531KC033525 | 20014    | 30002     | 80005    | 98345     |  12020.00 | 2020-01-31  |
| JH4KA4576KC031014 | 20011    | 30002     | 80002    | 98345     |  18020.00 | 2018-01-14  |
| JH4KA4630LC007479 | 20016    | 30002     | 80005    | 10101     |  72020.00 | 2012-01-31  |
| JH4KA4650JC000403 | 20028    | 30001     | 80004    | 10101     |  12020.00 | 2018-08-11  |
| JTDBE30K620061417 | 20009    | 30005     | 80009    | 76766     |  50020.00 | 2020-01-12  |
| LM4AC113061105688 | 20020    | 30001     | 80003    | 10101     | 203020.00 | 2017-10-21  |
| SAJWA0ES6DPS56028 | 20026    | 30005     | 80000    | 98345     |  32020.00 | 2010-06-11  |
| WAUDFAFL6DN014563 | 20011    | 30003     | 80002    | 10101     | 200020.00 | 2019-12-10  |
| WAUYGAFC6CN174200 | 20000    | 30000     | 80000    | 10101     |  80020.00 | 2020-01-11  |
```

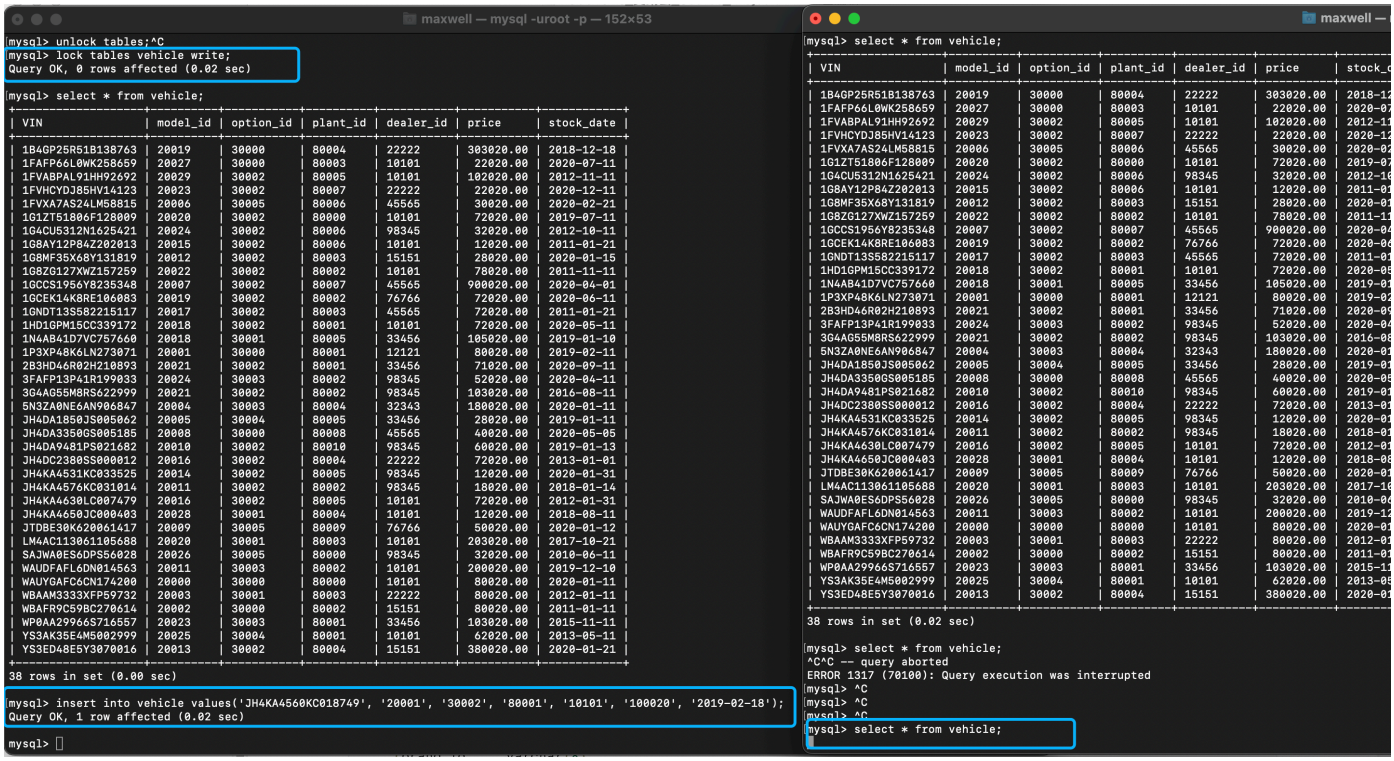
在sold表中有如下记录，对应的是每辆vehicle所出售的customer id和出售日期：

```
mysql> select * from sold;
+-------------------+-------------+------------+
| VIN               | customer_id | sale_date  |
+-------------------+-------------+------------+
| 1FAFP66L0WK258659 | 70557       | 2023-01-03 |
| 1FVABPAL91HH92692 | 98988       | 2023-01-03 |
| 1FVHCYDJ85HV14123 | 70557       | 2023-01-12 |
| 1FVXA7AS24LM58815 | 76653       | 2022-02-15 |
| 1G1ZT51806F128009 | 76653       | 2022-11-25 |
| 1G4CU5312N1625421 | 70557       | 2023-01-03 |
| 1G8AY12P84Z202013 | 76653       | 2023-01-03 |
| 1G8MF35X68Y131819 | 70557       | 2021-01-09 |
| 1G8ZG127XWZ157259 | 76653       | 2023-01-11 |
| 1GCCS1956Y8235348 | 98765       | 2022-01-05 |
| 1GCEK14K8RE106083 | 12345       | 2022-07-15 |
| 1GNDT13S582215117 | 98988       | 2022-03-05 |
| 1HD1GPM15CC339172 | 12345       | 2022-02-05 |
| 1P3XP48K6LN273071 | 19991       | 2022-04-02 |
| 2B3HD46R02H210893 | 12345       | 2022-11-05 |
| 3FAFP13P41R199033 | 54321       | 2023-01-03 |
| 5N3ZA0NE6AN906847 | 54321       | 2022-11-23 |
| JH4DA1850JS005062 | 70557       | 2022-12-24 |
| JH4DA3350GS005185 | 98988       | 2022-05-15 |
| JH4DA9481PS021682 | 55739       | 2021-01-10 |
| JH4DC2380SS000012 | 98988       | 2022-04-05 |
| JH4KA4531KC033525 | 70557       | 2023-01-02 |
| JH4KA4576KC031014 | 55739       | 2021-01-10 |
| JH4KA4630LC007479 | 12345       | 2023-01-05 |
| JH4KA4650JC000403 | 54321       | 2023-01-03 |
| JTDBE30K620061417 | 23121       | 2022-04-25 |
| SAJWA0ES6DPS56028 | 98988       | 2023-01-03 |
| WAUYGAFC6CN174200 | 12345       | 2021-04-25 |
| WBAAM3333XFP59732 | 45678       | 2022-10-11 |
| WBAFR9C59BC270614 | 44553       | 2021-09-15 |
| YS3AK35E4M5002999 | 76543       | 2023-01-03 |
| YS3ED48E5Y3070016 | 55739       | 2021-01-01 |
+-------------------+-------------+------------+
32 rows in set (0.04 sec)

mysql>
```

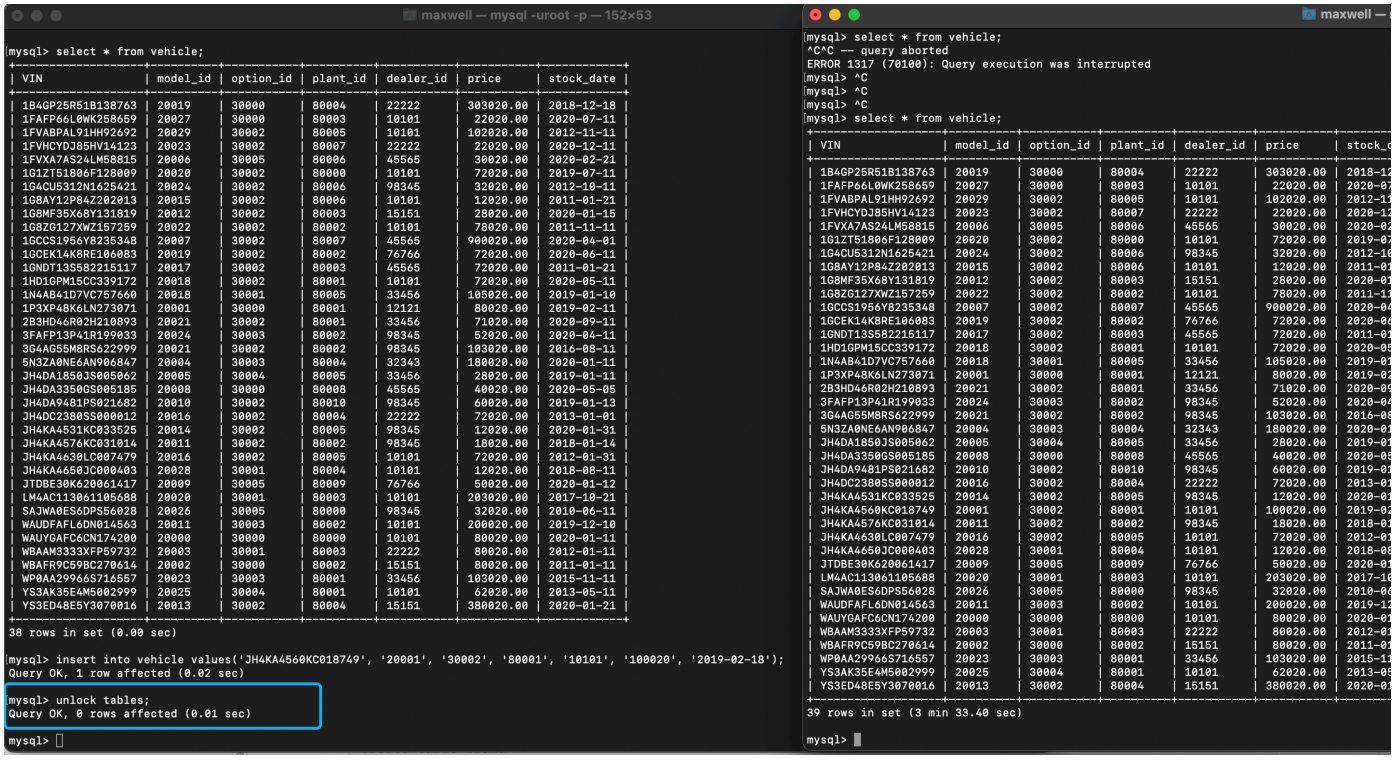
为了让多个用户并发操作该表时，避免其中有事务在修改该表时，其他事务读取该表时，产生读到脏数据或者不可重复读的情况；

首先为vehicle表设置写锁（即当前会话可以修改，查询表，其他会话将无法操作），再向vehicle表中插入新的记录：

```
lock tables vehicle write;

insert into vehicle values('JH4KA4560KC018749', '20001', '30002', '80001', '10101',
'100020', '2019-02-18');
```



在终端1为vehicle表添加写锁（或排他锁）之后，终端2中访问vehicle表会被阻塞，而终端1中可以对vehicle表进行正常读写操作；



终端1执行`unlock tables`之后，释放写锁，之后终端2可以正常读取vehicle中的数据，并且发现多了一条记录；

为vehicle表设置读锁后，当前会话和其他会话都不可以修改数据，但可以读取表数据（这里还需要为其引用外键的表上锁，否则会报错）

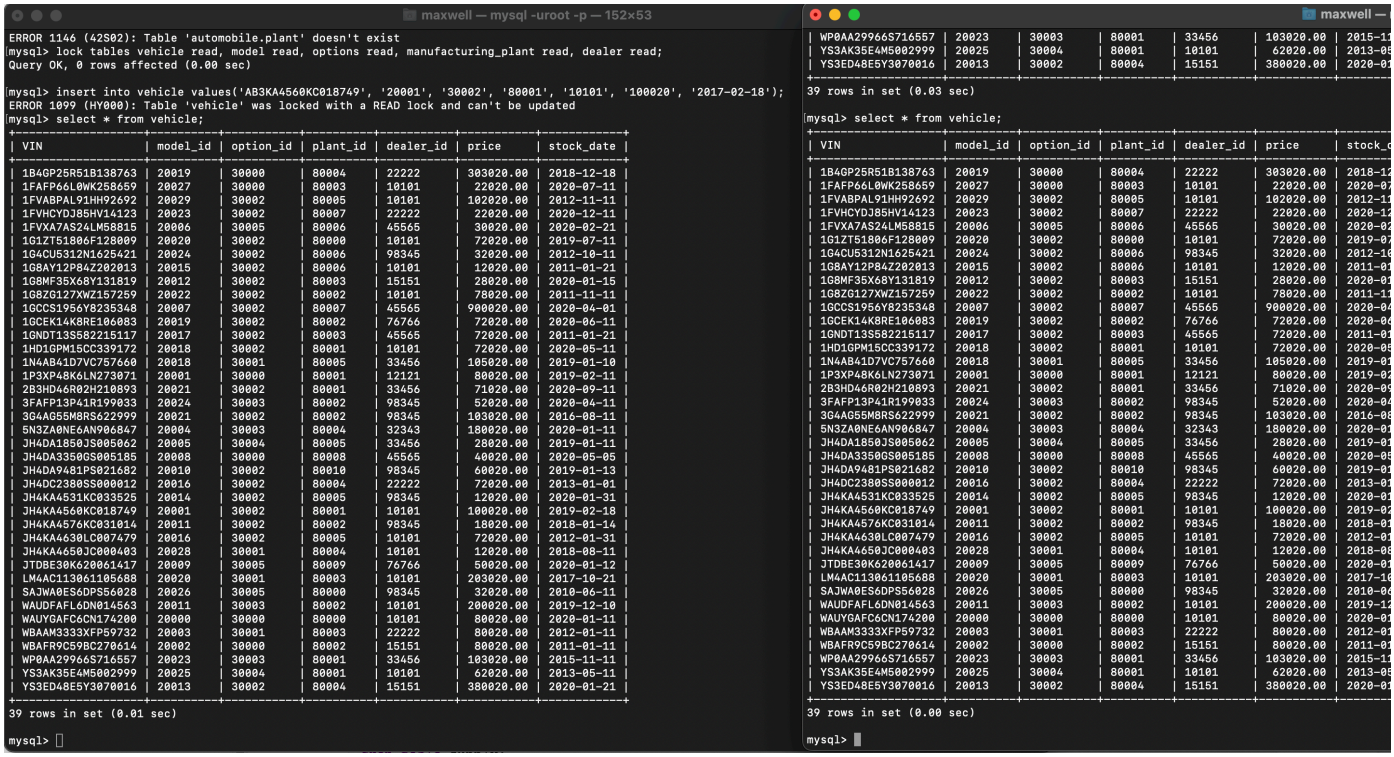
```
lock tables vehicle read, model read, options read, manufacturing_plant read, dealer read;
```

若在设置读锁的终端上对vehicle表进行插入：

```
insert into vehicle values('AB3KA4560KC018749', '20001', '30002', '80001', '10101', '100020', '2017-02-18');
```

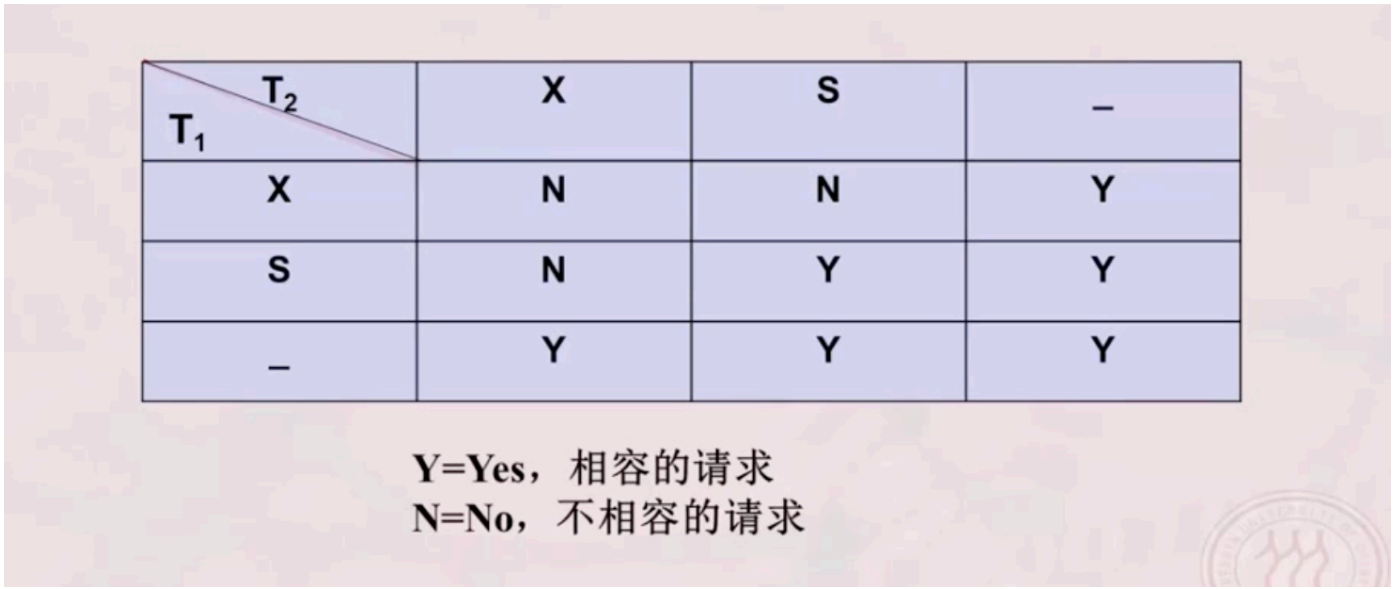
```
mysql> lock tables vehicle read, model read, options read, plant read, dealer read;
ERROR 1146 (42S02): Table 'automobile.plant' doesn't exist
mysql> lock tables vehicle read, model read, options read, manufacturing_plant read, dealer read;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into vehicle values('AB3KA4560KC018749', '20001', '30002', '80001', '10101', '100020', '2017-02-18');
ERROR 1099 (HY000): Table 'vehicle' was locked with a READ lock and can't be updated
mysql>
```

ERROR 1099 (HY000): Table 'vehicle' was locked with a READ lock and can't be updated表示当前会话不可以修改数据；



两个终端都可以读取数据；

> 锁的相容矩阵



- **三级封锁协议**
  在一级封锁协议之上，事务T在读取数据R之前必须先对其加S锁，直到事务结束才释放S锁。三级封锁协议除防止了丢失修改和不读“脏”数据外，还**进一步防止了不可重复读**。
- **四级封锁协议（对应serialization）**
  四级封锁协议是对三级封锁协议的增强，其实现机制也最为简单，直接对事务中所读取或者更改的数据所在的表**加表锁**，也就是说，其他事务不能读写该表中的任何数据。

如此，在该数据库中实现并发可以通过加锁并遵循四级封锁协议就可以实现多用户的并发操作；

VI. 相关代码

> DDL.sql

```
use Automobile;
-- 避免产生外键约束冲突，引用外键的表需要先于被引用的表删除

drop table sold;
drop table supply;
drop table vehicle;
drop table model;
drop table brand;
drop table supplier;
drop table customer;
drop table dealer;
drop table manufacturing_plant;
drop table options;
```

```
create table brand
    (brand_id     varchar(8),
     brand_name   varchar(20) not null,
      primary key (brand_id)
    );

create table model
    (model_id     varchar(8),
     brand_id     varchar(8),
     model_name   varchar(20) not null,
     body_style   varchar(20),
     primary key (model_id),
     foreign key (brand_id)   references brand (brand_id)
        on delete cascade
    );

create table options
    (option_id    varchar(8),
     color        varchar(20),
     engine       varchar(20),
     transmission varchar(20),
     primary key (option_id)
    );

create table dealer
    (dealer_id    varchar(8),
     dealer_name  varchar(20) not null,
     primary key (dealer_id)
    );

create table customer
    (customer_id      varchar(8),
     customer_name    varchar(20) not null,
     customer_address varchar(30),
     phone            varchar(20),
     gender           varchar(8),
     annual_income    numeric(8,2) check (annual_income >= 0),
     primary key (customer_id)
    );

create table manufacturing_plant
    (plant_id    varchar(8),
     plant_name  varchar(20) not null,
     part_name   varchar(20),
     assembler   bit,
     primary key (plant_id)
    );

create table supplier
```

```
    (supplier_id    varchar(8),
     supplier_name  varchar(20) not null,
     part_name      varchar(20),
     primary key (supplier_id)
    );

create table supply
    (supplier_id    varchar(8),
     model_id       varchar(8),
     primary key (supplier_id, model_id),
     foreign key (supplier_id) references supplier (supplier_id)
        on delete cascade,
     foreign key (model_id)   references model (model_id)
        on delete cascade
    );


create table vehicle
    (VIN        varchar(20),
     model_id   varchar(8),
     option_id  varchar(8),
     plant_id   varchar(8),
     dealer_id  varchar(8),
     price      numeric(8,2) check (price >= 0),
     stock_date date,
     primary key (VIN),
     foreign key (model_id)  references model (model_id)
        on delete cascade,
     foreign key (option_id) references options (option_id)
        on delete cascade,
     foreign key (plant_id)  references manufacturing_plant (plant_id)
        on delete cascade,
     foreign key (dealer_id) references dealer (dealer_id)
        on delete cascade
    );


create table sold
  (VIN              varchar(20),
     customer_id      varchar(8),
     sale_date        date,
   primary key (VIN, customer_id),
     foreign key (customer_id) references customer (customer_id)
        on delete cascade,
   foreign key (VIN) references vehicle (VIN)
        on delete cascade
    );
```

RelationInsertFile.sql

```
use Automobile;
delete from brand;
delete from customer;
delete from dealer;
delete from manufacturing_plant;
delete from model;
delete from options;
delete from sold;
delete from supplier;
delete from supply;
delete from vehicle;

insert into brand values ('10000', 'Audi');
insert into brand values ('10001', 'BMW');
insert into brand values ('10002', 'FORD');
insert into brand values ('10003', 'Porsche');
insert into brand values ('10004', 'Tesla');
insert into brand values ('10005', 'Kia');
insert into brand values ('10006', 'Peugeot');
insert into brand values ('10007', 'Honda');
insert into brand values ('10008', 'Jaguar');
insert into brand values ('10009', 'Mazda');

insert into model values ('20000', '10000','SQ8 e-tron', 'SUV');
insert into model values ('20001', '10000','TT', 'Cabriolet');
insert into model values ('20002', '10000','S1', 'Hatchback');
insert into model values ('20003', '10001','V8', 'Sedan');
insert into model values ('20004', '10001','PB18', 'Coupe');
insert into model values ('20005', '10001','M8', 'Coupe');
insert into model values ('20006', '10002','8 Series', 'Coupe');
insert into model values ('20007', '10002','XM', 'SUV');
insert into model values ('20008', '10002','Z3', 'Cabriolet');
insert into model values ('20009', '10003','X4', 'SAC');
insert into model values ('20010', '10003','Thunderbird', 'Cabriolet');
insert into model values ('20011', '10003','Puma', 'Coupe');
insert into model values ('20012', '10004','Explorer', 'SUV');
insert into model values ('20013', '10004','Escape', 'SUV');
insert into model values ('20014', '10004','Focus', 'Station wagon');
insert into model values ('20015', '10005','718', 'Cabriolet');
insert into model values ('20016', '10005','911', 'Coupe');
insert into model values ('20017', '10005','917', 'SUV');
insert into model values ('20018', '10006','Cayenne', 'SUV');
insert into model values ('20019', '10006','Cayman', 'Coupe');
```

```
insert into model values ('20020', '10006','718', 'Cabriolet');
insert into model values ('20021', '10007','911', 'Coupe');
insert into model values ('20022', '10007','917', 'SUV');
insert into model values ('20023', '10007','Cayenne', 'SUV');
insert into model values ('20024', '10008','Cayman', 'Coupe');
insert into model values ('20025', '10008','E-Type', 'Cabriolet');
insert into model values ('20026', '10008','S-Type', 'SUV');
insert into model values ('20027', '10009','MX-5', 'Coupe');
insert into model values ('20028', '10009','CX-50', 'SUV');
insert into model values ('20029', '10009','CX-30', 'SUV');

insert into options values('30000', 'Amplify Orange', 'Thermal', 'Automatic');
insert into options values('30001', 'Stryker Red', 'Electrical', 'Automatic');
insert into options values('30002', 'Forged Green', 'Combustion', 'Manual');
insert into options values('30003', 'Diffused Sky Blue', 'Twin-Turbo', 'Automatic');
insert into options values('30004', 'Digital Teal', 'Hybrid', 'CVT');
insert into options values('30005', 'Earl Gray', 'Electrical', 'Semi-automatic');

insert into dealer values ('10101', 'Srinivasan');
insert into dealer values ('12121', 'Wu');
insert into dealer values ('15151', 'Mozart');
insert into dealer values ('22222', 'Einstein');
insert into dealer values ('32343', 'El Said');
insert into dealer values ('33456', 'Gold');
insert into dealer values ('45565', 'Katz');
insert into dealer values ('58583', 'Califieri');
insert into dealer values ('76543', 'Singh');
insert into dealer values ('76766', 'Crick');
insert into dealer values ('83821', 'Brandt');
insert into dealer values ('98345', 'Kim');

insert into customer values ('12345', 'Shankar', 'Nottingham', '3210000', 'F',
'90000');
insert into customer values ('19991', 'Brandt', 'New York', '8010003', 'F', '90000');
insert into customer values ('23121', 'Chavez', 'Toronto', '1110001', 'M', '90000');
insert into customer values ('44553', 'Peltier', 'Los Angeles', '5610004', 'F',
'120000');
insert into customer values ('45678', 'Levy', 'Montreal', '4610001', 'F', '90000');
insert into customer values ('54321', 'Williams', 'Havana', '5410009', 'M', '90000');
insert into customer values ('55739', 'Sanchez', 'Philadelphia', '3810010', 'F',
'90000');
insert into customer values ('70557', 'Snow', 'Chicago', '0100122', 'F', '70000');
insert into customer values ('76543', 'Brown', 'Guatemala City', '5810002', 'F',
'200000');
insert into customer values ('76653', 'Aoi', 'Havana', '6010003', 'M', '40000');
insert into customer values ('98765', 'Bourikas', 'Nottingham', '9810010', 'F',
'190000');
insert into customer values ('98988', 'Tanaka', 'Beijing', '1201000', 'M', '20000');
```

```
insert into supplier values('50000', 'Ketai Industries', 'door');
insert into supplier values('50001', 'Zhengde Auto Parts', 'seat');
insert into supplier values('50002', 'Fronte Motor Parts', 'engine');
insert into supplier values('50003', 'Getrag', 'transmission');
insert into supplier values('50004', 'Allied Auto Stores', 'chassis');
insert into supplier values('50005', 'Central Parts Perth', 'engine');
insert into supplier values('50006', 'Kei Industries', 'battery');
insert into supplier values('50007', 'Wiki Industries', 'transmission');
insert into supplier values('50008', 'IILi Industries', 'door');
insert into supplier values('50009', 'PLK Industries', 'seat');

insert into supply values('50000', '20000');
insert into supply values('50001', '20001');
insert into supply values('50002', '20002');
insert into supply values('50003', '20003');
insert into supply values('50004', '20004');
insert into supply values('50005', '20005');
insert into supply values('50006', '20006');
insert into supply values('50007', '20007');
insert into supply values('50008', '20008');
insert into supply values('50009', '20009');
insert into supply values ('50007', '20010');
insert into supply values ('50007','20011');
insert into supply values ('50001','20012');
insert into supply values ('50000','20013');
insert into supply values ('50002','20014');
insert into supply values ('50003','20015');
insert into supply values ('50004','20016');
insert into supply values ('50005','20017');
insert into supply values ('50006','20018');
insert into supply values ('50001','20019');
insert into supply values ('50002','20020');
insert into supply values ('50003','20021');
insert into supply values ('50004','20022');
insert into supply values ('50005','20023');
insert into supply values ('50007','20024');


insert into manufacturing_plant values ('80000', 'Pothos', 'door', 0);
insert into manufacturing_plant values ('80001', 'Mint', 'seat', 0);
insert into manufacturing_plant values ('80002', 'Cacti', 'engine', 0);
insert into manufacturing_plant values ('80003', 'Bromeliads', NULL, 1);
insert into manufacturing_plant values ('80004', 'Daffodil', NULL, 1);
insert into manufacturing_plant values ('80005', 'Mayflower', NULL, 1);
insert into manufacturing_plant values ('80006', 'Laurel', NULL, 1);
insert into manufacturing_plant values ('80007', 'Rosa', NULL, 1);
insert into manufacturing_plant values ('80008', 'Plant1', NULL, 1);
insert into manufacturing_plant values ('80009', 'Plant2', 'engine', 0);
insert into manufacturing_plant values ('80010', 'Plant3', NULL, 1);
```

```
insert into vehicle values('WAUYGAFC6CN174200', '20000', '30000', '80000', '10101',
'80020', '2020-01-11');
insert into vehicle values('1P3XP48K6LN273071', '20001', '30000', '80001', '12121',
'80020', '2019-02-11');
insert into vehicle values('WBAFR9C59BC270614', '20002', '30000', '80002', '15151',
'80020', '2011-01-11');
insert into vehicle values('WBAAM3333XFP59732', '20003', '30001', '80003', '22222',
'80020', '2012-01-11');
insert into vehicle values('5N3ZA0NE6AN906847', '20004', '30003', '80004', '32343',
'180020', '2020-01-11');
insert into vehicle values('JH4DA1850JS005062', '20005', '30004', '80005', '33456',
'28020', '2019-01-11');
insert into vehicle values('1FVXA7AS24LM58815', '20006', '30005', '80006', '45565',
'30020', '2020-02-21');
insert into vehicle values('1GCCS1956Y8235348', '20007', '30002', '80007', '45565',
'900020', '2020-04-01');
insert into vehicle values('JH4DA3350GS005185', '20008', '30000', '80008', '45565',
'40020', '2020-05-05');
insert into vehicle values('JTDBE30K620061417', '20009', '30005', '80009', '76766',
'50020', '2020-01-12');
insert into vehicle values('JH4DA9481PS021682', '20010', '30002', '80010', '98345',
'60020', '2019-01-13');
insert into vehicle values('JH4KA4576KC031014', '20011', '30002', '80002', '98345',
'18020', '2018-01-14');
insert into vehicle values('1G8MF35X68Y131819', '20012', '30002', '80003', '15151',
'28020', '2020-01-15');
insert into vehicle values('YS3ED48E5Y3070016', '20013', '30002', '80004', '15151',
'380020', '2020-01-21');
insert into vehicle values('JH4KA4531KC033525', '20014', '30002', '80005', '98345',
'12020', '2020-01-31');
insert into vehicle values('1G8AY12P84Z202013', '20015', '30002', '80006', '10101',
'12020', '2011-01-21');
insert into vehicle values('JH4KA4630LC007479', '20016', '30002', '80005', '10101',
'72020', '2012-01-31');
insert into vehicle values('JH4DC2380SS000012', '20016', '30002', '80004', '22222',
'72020', '2013-01-01');
insert into vehicle values('1GNDT13S582215117', '20017', '30002', '80003', '45565',
'72020', '2011-01-21');
insert into vehicle values('1HD1GPM15CC339172', '20018', '30002', '80001', '10101',
'72020', '2020-05-11');
insert into vehicle values('1GCEK14K8RE106083', '20019', '30002', '80002', '76766',
'72020', '2020-06-11');
insert into vehicle values('1G1ZT51806F128009', '20020', '30002', '80000', '10101',
'72020', '2019-07-11');
insert into vehicle values('2B3HD46R02H210893', '20021', '30002', '80001', '33456',
'71020', '2020-09-11');
insert into vehicle values('1G8ZG127XWZ157259', '20022', '30002', '80002', '10101',
'78020', '2011-11-11');
```

```
insert into vehicle values('1FVHCYDJ85HV14123', '20023', '30002', '80007', '22222',
'22020', '2020-12-11');
insert into vehicle values('1G4CU5312N1625421', '20024', '30002', '80006', '98345',
'32020', '2012-10-11');
insert into vehicle values('3FAFP13P41R199033', '20024', '30003', '80002', '98345',
'52020', '2020-04-11');
insert into vehicle values('YS3AK35E4M5002999', '20025', '30004', '80001', '10101',
'62020', '2013-05-11');
insert into vehicle values('SAJWA0ES6DPS56028', '20026', '30005', '80000', '98345',
'32020', '2010-06-11');
insert into vehicle values('1FAFP66L0WK258659', '20027', '30000', '80003', '10101',
'22020', '2020-07-11');
insert into vehicle values('JH4KA4650JC000403', '20028', '30001', '80004', '10101',
'12020', '2018-08-11');
insert into vehicle values('1FVABPAL91HH92692', '20029', '30002', '80005', '10101',
'102020', '2012-11-11');

insert into sold values('WAUYGAFC6CN174200', '12345', '2021-04-25');
insert into sold values('1P3XP48K6LN273071', '19991', '2022-04-02');
insert into sold values('WBAFR9C59BC270614', '44553', '2021-09-15');
insert into sold values('WBAAM3333XFP59732', '45678', '2022-10-11');
insert into sold values('5N3ZA0NE6AN906847', '54321', '2022-11-23');
insert into sold values('JH4DA1850JS005062', '70557', '2022-12-24');
insert into sold values('1FVXA7AS24LM58815', '76653', '2022-02-15');
insert into sold values('1GCCS1956Y8235348', '98765', '2022-01-05');
insert into sold values('JH4DA3350GS005185', '98988', '2022-05-15');
insert into sold values('JTDBE30K620061417', '23121', '2022-04-25');
insert into sold values('JH4DA9481PS021682', '55739', '2021-01-10');
insert into sold values('JH4KA4576KC031014', '55739', '2021-01-10');
insert into sold values('1G8MF35X68Y131819', '70557', '2021-01-09');
insert into sold values('YS3ED48E5Y3070016', '55739', '2021-01-01');
insert into sold values('JH4KA4531KC033525', '70557', '2023-01-02');
insert into sold values('1G8AY12P84Z202013', '76653', '2023-01-03');
insert into sold values('JH4KA4630LC007479', '12345', '2023-01-05');
insert into sold values('JH4DC2380SS000012', '98988', '2022-04-05');
insert into sold values('1GNDT13S582215117', '98988', '2022-03-05');
insert into sold values('1HD1GPM15CC339172', '12345', '2022-02-05');
insert into sold values('1GCEK14K8RE106083', '12345', '2022-07-15');
insert into sold values('1G1ZT51806F128009', '76653', '2022-11-25');
insert into sold values('2B3HD46R02H210893', '12345', '2022-11-05');
insert into sold values('1G8ZG127XWZ157259', '76653', '2023-01-11');
insert into sold values('1FVHCYDJ85HV14123', '70557', '2023-01-12');
insert into sold values('1G4CU5312N1625421', '70557', '2023-01-03');
insert into sold values('3FAFP13P41R199033', '54321', '2023-01-03');
insert into sold values('YS3AK35E4M5002999', '76543', '2023-01-03');
insert into sold values('SAJWA0ES6DPS56028', '98988', '2023-01-03');
insert into sold values('JH4KA4650JC000403', '54321', '2023-01-03');
insert into sold values('1FAFP66L0WK258659', '70557', '2023-01-03');
insert into sold values('1FVABPAL91HH92692', '98988', '2023-01-03');
```

```
update model
set body_style = 'Convertible'
where body_style = 'Cabriolet';
```