Review Questions:

1. When storing dates as the date datatype it becomes easier to manipulate that data when retrieving it. Data stored as date can use all the functions associated with date. Date can be presented in any date format in a query, a character would have to be parsed into a date and then represented as the correct date format. It would be worse for everyone, the people who enter data into a database and those who query it.

7. Where V_State = 'TN' OR V_State = 'FL' OR V_State = 'GA'

9. The first command would produce one line, the count of V_CODE entries in PRODUCT. The distinct would not matter here because count only returns one result, the count of V_CODE in PRODUCT.

The second command would count only the first V_CODE in PROCUT that have that distinct V_CODE. It would return a count of the entries in a list of V_CODE with no duplicate V_CODES that were in PRODUCT.

11. WHERE in SQL is a condition that is applied while the query is being executed. It will only return results that fit the criteria of the WHERE clause. The HAVING condition by contract only filters the results that have been returned after the interaction with the server is finished and is used after the results have been grouped.

Problems:

9.

```
select count(INV_NUMBER) as `Number of Invoices`

      from INVOICE;
```

10.

```
select count(CUS_CODE) as `Number of customers who have a balance over 500$`

      from CUSTOMER

    where CUS_BALANCE > 500;
```

12.

```
select CUS_CODE, INV_NUMBER, P_DESCRIPT, LINE_UNIT as `Units bought`,
LINE_PRICE as `Unit Price`, TRUNCATE(LINE_UNIT * LINE_PRICE, 2) as `Subtotal`

      from CUSTOMER

    natural join LINE

    natural join PRODUCT

    natural join INVOICE

    order by CUS_CODE, INV_NUMBER, P_DESCRIPT;
```

15.

```
select CUS_CODE, CUS_BALANCE, SUM(LINE_UNIT * LINE_PRICE) as `Total
Purchases` , COUNT(LINE_NUMBER) as `Number of Purchases`,
TRUNCATE(SUM(LINE_UNIT * LINE_PRICE)/COUNT(LINE_NUMBER), 2) as `Average
Purchase Amount`

      from CUSTOMER

    natural join LINE

    natural join INVOICE

    group by CUS_CODE

    order by CUS_CODE;
```

23.

```
select CUSTOMER.CUS_CODE, CUS_BALANCE

from

    CUSTOMER

left join

    INVOICE on CUSTOMER.CUS_CODE = INVOICE.CUS_CODE

where

    INV_NUMBER is null

order by CUS_CODE;
```

27.

```
select *

      from LGDEPARTMENT

    order by DEPT_NAME;
```

28.

```
select PROD_SKU, PROD_DESCRIPT, PROD_TYPE, PROD_BASE, PROD_CATEGORY,
PROD_PRICE

      from LGPRODUCT

    where PROD_BASE = 'Water' and PROD_CATEGORY = 'Sealer';
```

32.

```
select CUST_FNAME, CUST_LNAME, CUST_STREET, CUST_CITY, CUST_STATE, CUST_ZIP

      from LGBRAND

      natural join LGPRODUCT

      natural join LGINVOICE

      natural join LGLINE

      natural join LGCUSTOMER

where BRAND_NAME = 'FORESTERS BEST' and PROD_CATEGORY = 'Top Coat' and
INV_DATE between '2017-7-15' and '2017-7-31'

group by CUST_STATE, CUST_LNAME, CUST_FNAME

order by CUST_STATE, CUST_LNAME, CUST_FNAME;
```