Problem 6.4.1 (15 points)

Part b) (5 points)

```sql
SELECT DISTINCT maker
FROM Product, Laptop
WHERE Product.model = Laptop.model AND hd >= 1;
```

Part c) (5 points)

```sql
(SELECT DISTINCT PC.model as model,
     PC.price as price
FROM Product, PC
WHERE Product.model = PC.model AND maker = 'B')
UNION
(SELECT DISTINCT Laptop.model as model,
     Laptop.price as price
FROM Product, Laptop
WHERE Product.model = Laptop.model AND maker = 'B')
UNION
(SELECT DISTINCT Printer.model as model,
     Printer.price as price
FROM Product, Printer
WHERE Product.model = Printer.model AND maker = 'B');
```

Part d) (5 points)

```sql
SELECT DISTINCT model
FROM Printer
WHERE color = 'true' AND type = 'laser';
```
Problem 6.4.6 (30 points)

Part b) (5 points)

SQL: SELECT AVG(speed) 
    FROM Laptop 
    WHERE price > 2000;

Result: 

<table>
<thead>
<tr>
<th>AVG(speed)</th>
<th>775</th>
</tr>
</thead>
</table>

Part c) (5 points)

SQL: SELECT AVG(price) 
    FROM Product, PC 
    WHERE Product.model = PC.model AND maker = 'A';

Result: 

<table>
<thead>
<tr>
<th>AVG(price)</th>
<th>1776</th>
</tr>
</thead>
</table>

Part e) (5 points)

SQL: SELECT speed, AVG(price) 
    FROM PC 
    GROUP BY speed;

Result: | speed | AVG(price) |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>700</td>
<td>899</td>
</tr>
<tr>
<td>1500</td>
<td>2499</td>
</tr>
<tr>
<td>866</td>
<td>1499</td>
</tr>
<tr>
<td>1000</td>
<td>1499</td>
</tr>
<tr>
<td>1300</td>
<td>2199</td>
</tr>
<tr>
<td>1400</td>
<td>2299</td>
</tr>
<tr>
<td>1200</td>
<td>1699</td>
</tr>
<tr>
<td>750</td>
<td>699</td>
</tr>
<tr>
<td>1100</td>
<td>1299</td>
</tr>
<tr>
<td>350</td>
<td>799</td>
</tr>
<tr>
<td>733</td>
<td>2499</td>
</tr>
</tbody>
</table>
Part g)  (5 points)

SQL: SELECT maker
    FROM Product
    WHERE type = 'PC'
    GROUP BY maker
    HAVING COUNT(model) >= 3;

Result:

<table>
<thead>
<tr>
<th>maker</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>D</td>
</tr>
</tbody>
</table>

Part h)  (5 points)

SQL: SELECT maker, MAX(price)
    FROM Product, PC
    WHERE Product.model = PC.model
    GROUP BY maker;

Result:

<table>
<thead>
<tr>
<th>maker</th>
<th>MAX(price)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2499</td>
</tr>
<tr>
<td>B</td>
<td>2119</td>
</tr>
<tr>
<td>C</td>
<td>2299</td>
</tr>
<tr>
<td>D</td>
<td>1699</td>
</tr>
<tr>
<td>E</td>
<td>2499</td>
</tr>
</tbody>
</table>

Part i)  (5 points)

SQL: SELECT speed, AVG(price)
    FROM PC
    WHERE speed > 800
    GROUP BY speed;

Result:

<table>
<thead>
<tr>
<th>speed</th>
<th>AVG(price)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1500</td>
<td>2499</td>
</tr>
<tr>
<td>866</td>
<td>1499</td>
</tr>
<tr>
<td>1000</td>
<td>1499</td>
</tr>
<tr>
<td>1300</td>
<td>2199</td>
</tr>
<tr>
<td>1400</td>
<td>2299</td>
</tr>
<tr>
<td>1200</td>
<td>1699</td>
</tr>
<tr>
<td>1100</td>
<td>1299</td>
</tr>
</tbody>
</table>
Problem 6.5.1  (20 points)

Part a)  (5 points)

SQL 1:  
\[
\text{INSERT INTO Product} \\
\text{VALUES(‘C’, 1100, ‘pc’);}
\]

SQL 2:  
\[
\text{INSERT INTO PC} \\
\text{VALUES(1100, 1800, 256, 80, ‘20x DVD’, 2499);}
\]

Effect 1:  Product relation now has a new tuple with the above attributes.

Effect 2:  PC table now has a corresponding tuple describing in further detail about model 1100, such as the speed, hd, rd, etc.

Part d)  (5 points)

SQL:
\[
\text{DELETE FROM Laptop} \\
\text{WHERE Laptop.model IN} \\
\text{(SELECT model} \\
\text{FROM Product} \\
\text{WHERE maker NOT IN} \\
\text{(SELECT maker} \\
\text{FROM Product} \\
\text{WHERE type = ‘printer’))};
\]

Effect: From the Laptop relation, all tuples that have models corresponding to makers that make printers are deleted.

Part e)  (5 points)

SQL:
\[
\text{UPDATE Product} \\
\text{SET maker = ‘A’} \\
\text{WHERE maker = ‘B’};
\]

Effect: For the Product table, all makers that were ‘B’ are now ‘A’.

Part f)  (5 points)

SQL:
\[
\text{UPDATE PC} \\
\text{SET ram = ram * 2,} \\
\text{hd = hd + 20;}
\]

Effect: For every entry in PC, all RAM values are now doubled (x2) and all hard-drive values have gained 20 gigabytes.
Problem 6.5.2   (5 points)

Part e)   (5 points)

SQL:   
DELETES FROM Classes  
WHERE class IN  
(SELECT class  
FROM Ships  
GROUP BY class  
HAVING COUNT(name) < 3);  

Effect:   From the Classes table, all tuples whose class attribute corresponded to having fewer than 3 ships were deleted.
Problem 6.6.2  (25 points)

Part a)  (5 points)

CREATE TABLE Product
(
    maker CHAR(1),
    model INTEGER,
    type VARCHAR(7)
);

Part b)  (5 points)

CREATE TABLE PC
(
    model INTEGER,
    speed INTEGER,
    ram INTEGER,
    hd INTEGER,
    rd VARCHAR(10),
    price INTEGER
);

Part d)  (5 points)

CREATE TABLE Printer
(
    model INTEGER,
    color BOOLEAN,
    type VARCHAR(7),
    price INTEGER
);

Part e)  (5 points)

ALTER TABLE Printer DROP color;

Part f)  (5 points)

ALTER TABLE Laptop ADD cd VARCHAR(10) DEFAULT 'none';
Problem 6.7.1  (10 points)

Part b)  (5 points)

CREATE VIEW StudioPres AS
    SELECT   MovieExec.name as name,
            MovieExec.address as address,
            cert#
    FROM Studio, MovieExec
    WHERE cert# = presC#;

Part c)  (5 points)

CREATE VIEW ExecutiveStar AS
    SELECT   MovieStar.name as name,
            MovieStar.address as address,
            gender,
            birthdate,
            cert#,
            networth
    FROM MovieStar, MovieExec
    WHERE MovieStar.name = MovieExec.name
        AND MovieStar.address = MovieExec.address;
Problem 6.7.2 (10 points)

Part a) (5 points)

**Updatable**
We are only grabbing attributes from one table and so long as we insert the necessary or required information (all attributes displayed), then we can update the view. Also note that RichExec is not part of any subquery of the original view creation:

```
CREATE VIEW RichExec AS
    SELECT *
    FROM MovieExec
    WHERE networth >= 1000000;
```

Part c) (5 points)

**Not Updatable**
Even if we insert all the necessary information (all the displayed attributes), either:

```
MovieStar.name and MovieStar.address
```

or

```
MovieExec.name and MovieExec.address
```

will be NULL (in my case, the latter). Thus, there will be no way to compare the names and addresses in the WHERE clause, causing updating problems.
Problem 6.7.4  

(6 points)

Part a)  

(2 points)

\[ \pi_{\text{title}, \text{name}} \]

\[ \sigma_{\text{producerC#} = \text{cert#}} \]

\[ \times \]

Movie  MovieExec

Part b)  

(2 points)

\[ \pi_{\text{name}} \]

\[ \sigma_{\text{title} = '\text{Gone With the Wind}' } \]

MovieProd

Part c)  

(2 points)

\[ \pi_{\text{name}} \]

\[ \sigma_{\text{title} = '\text{Gone With the Wind}' } \]

\[ \pi_{\text{title}, \text{name}} \]

\[ \sigma_{\text{producerC#} = \text{cert#}} \]

\[ \times \]

Movie  MovieExec