一、

1)

```
SELECT CNAME, TEACHER
FROM S, C, SC
WHERE S.S# = SC.S# AND C.C# = SC.C# AND S.S# = "003";
```

2)

```
SELECT DISTINCT SNAME
FROM S, C, SC
WHERE S.S# = SC.S# AND C.C# = SC.C# AND S.AGE = " 男"
     AND C.TEACHER = " 程军";
```

3)

```
SELECT CNAME
FROM C
WHERE CNAME NOT IN(
    SELECT DISTINCT CNAME
    FROM S, C, SC
    WHERE S.S# = SC.S# AND C.C# = SC.C# AND S.SNAME = " 刘丽";
)
```

4)

```
SELECT S#, SNAME
FROM S, SC
WHERE S.S# = SC.S#
GROUP BY S.S# HAVING AVG(GRADE) < 60;
```

注意此处的问题，在使用了 GROUP BY 子句后，SELECT 子句的列名表中只能出现分组属性和聚集函数；从而，因为因为此处需要 S#，SNAME，从而在分组时应该同时选择 S#，SNAME，修改后的 SQL 语句如下：

```
SELECT S#, SNAME
FROM S, SC
WHERE S.S# = SC.S#
GROUP BY S.S# S.SNAME HAVING AVG(GRADE) < 60;
```

5)

```
SELECT S#
FROM S, SC
WHERE S.S# = SC.S#
GROUP BY S.S# HAVING COUNT(*) >= 3
ORDER BY S# ASC;
```

二、

1)

```
CREATE TABLE Classes
(
    class CHAR(20) PRIMARY KEY,
    type CHAR(2) NOT NULL,
    country CHAR(15),
    numGuns SMALLINT,
    bore, SMALLINT,
    displacement, INT
);

CREATE TABLE Ships
(
    name CHAR(20) PRIMARY KEY,
    class CHAR(20),
    launched SMALLINT,
    FOREIGN KEY (class) REFERENCES Classes(class)
);
```

注意

- 在插入 Classes 时，主键 class 可以唯一区分实体;
- 在插入 Ships 时，主键 name 可以唯一区分实体，并且外键 class 需要存在。

2) a.

```
INSERT
INTO Classes
VALUES("Nelson", "bb", "Gt.Britain", 9, 16, 34000);

INSERT
INTO Ships
VALUES("Nelson", "Nelson", 1927);

INSERT
INTO Ships
VALUES("Rodney", "Nelson", 1927);
```

b.

```
DELETE
FROM Ships
WHERE name IN
(
    SELECT
    FROM Ships, Outcomes
    WHERE Ships.name = Outcomes.ship
        AND Outcomes.result = "sunk"
);
```

c.
```
UPDATE Classes
SET bore = bore * 2.5;

UPDATE Classes
SET displacement = displacement / 1.1;
```

3)
```
CREATE VIEW (class, type, numGuns, bore, displacement, launched)
AS
SELECT class, type, numGuns, bore, displacement, launched
FROM Classes, Ships
WHERE Classes.class = Ships.ship
    AND country = "Gt.Britain";
```

4)
```
SELECT battle
FROM Ships, Outcomes
WHERE Ships.name = Outcomes.ship
    AND Ships.class = "Kongo";
```

5)
```
SELECT AVG(numGuns)
FROM Classes
WHERE type = "bb";
```

注意此处的问题应该是，所有的主力舰的火炮数量加和然后平均，改正后的 SQL 语句如下：

```
SELECT AVG(numGuns)
FROM Classes, Ships
WHERE Classes.class = Ships.class AND Classes.type = "bb";
```

6)
```
SELECT  MIN(launched)
FROM Classes, Ships
WHERE Classes.class = Ships.class
GROUP BY Classes.class;
```

注意需要同时显示类名和下水年份，修改后的 SQL 语句如下：

```
SELECT  Classes.class, MIN(launched)
FROM Classes, Ships
WHERE Classes.class = Ships.class
GROUP BY Classes.class;
```

7) **下面这种写法存在问题，对于没有船沉没但是有超过 3 艘船的类型，没有输出！**

```
SELECT class, COUNT(result)
FROM Ships, Outcomes
```

```
WHERE Ships.name = Outcomes.ship
    AND Outcomes.result = "sunk"
    AND class IN
    (
        SELECT class
        FROM Ships, Outcomes
        WHERE Ships.name = Outcomes.ship
            GROUP BY class HAVING COUNT(*) >= 3
    )
    GROUP BY class;
```

应该改为如下写法，注意不使用 where 过滤是否沉船条件：

```
SELECT class, SUM(case 1 when Outcomes.result = "sunk" else 0 end)
FROM Ships, Outcomes
WHERE Ships.name = Outcomes.ship
    AND class IN
    (
        SELECT class
        FROM Ships, Outcomes
        WHERE Ships.name = Outcomes.ship
            GROUP BY class HAVING COUNT(*) >= 3
    )
    GROUP BY class;
```

8)

```
SELECT name
FROM Ships
UNION
SELECT ship
FROM Outcomes;
```

9)

```
SELECT country
FROM Classes
GROUP BY country HAVING COUNT(type) >= 2;
```

10)

```
SELECT country
FROM Classes
WHERE numGuns >= ALL
    (
        SELECT numGuns
        FROM Classes
    );
```

11)

```
SELECT Classes.class
FROM Classes, Ships, Outcomes
WHERE Classes.class = Ships.class
    AND Ships.name = Outcomes.ship
    AND Outcomes.result = "sunk"
GROUP BY Classes.class HAVING COUNT(*) >= 1;
```

三、

1)

```
CREATE TABLE Orders
(
    OrderID CHAR(20) PRIMARY KEY,
    SupplierID CHAR(20),
    MovieID CHAR(20),
    Copies INT,
    FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID),
    FOREIGN KEY (MovieID) REFERENCES Movies(MovieID)
);

CREATE TABLE Rentals
(
    CustomerID CHAR(20),
    TapeID CHAR(20),
    CkoutDate DATE,
    Duration INT,
    PRIMARY KEY (CustomerID, TapeID, CkoutDate),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
    FOREIGN KEY (TapeID) REFERENCES Inventory(TapeID)
);
```

2) a.

```
INSERT
INTO Rentals
VALUES("9823", "5600", 2017-03-26, 30);
```

b.

```
DELETE
FROM Rentals
WHERE CkoutDate < '2000-01-01';
```

c.

```
UPDATE MovieSupplier
SET Price = Price / 6.88;
```

3)

```
CREATE VIEW JHV_Suppliers (MovieName, Price)
AS
```

```
SELECT MovieName, Price
FROM Movies, Suppliers, MovieSupplier
WHERE Movies.MovieID = MovieSupplier.MovieID
    AND Suppliers.SupplierID = MovieSupplier.SupplierID
    AND Suppliers.SupplierName = "Joe's House of Video";
```

4)

```
SELECT SupplierName, COUNT(MovieID)
FROM Inventory, Suppliers, MovieSupplier
WHERE Inventory.MovieID = MovieSupplier.MovieID
    AND Suppliers.SupplierID = MovieSupplier.SupplierID
GROUP BY Suppliers.SupplierID;
```

5)

```
SELECT MovieName
FROM Orders, Movies
WHERE Movies.MovieID = Orders.MovieID
    AND SUM(Orders.Copies) > 5
GROUP BY Orders.MovieID;
```

6)

```
SELECT MovieName
FROM Inventory, Movies
WHERE Movies.MovieID = Inventory.MovieID
GROUP BY Movies.MovieID HAVING COUNT(TapeID) > 1;
```

7)

```
SELECT MovieName
FROM Movies, Inventory, Rentals
WHERE Movies.MovieID = Inventory.MovieID
    AND Rentals.TapeID = Inventory.TapeID
    AND Rentals.Duration >= ALL
    (
        SELECT Duration
        FROM Rentals
    );
```

8)

```
SELECT DISTINCT MovieName
FROM Movies
WHERE MovieName NOT IN(
    SELECT DISTINCT MovieName
    FROM Movies, Inventory
    WHERE Movies.MovieID = Inventory.MovieID;
)
```

9)

```
SELECT SupplierID, SupplierName
FROM Suppliers, MovieSupplier
```

```
WHERE Suppliers.SupplierID = MovieSupplier.MovieID
    AND MovieSupplier.MovieID IN
    (
        SELECT MovieID
        FROM Movies
        WHERE MovieName = "Hacksaw Ridge"
    )
    AND MovieSupplier.Price <= ALL
    (
        SELECT Price
        FROM Movies, MovieSupplier
        WHERE Movies.MovieID = MovieSupplier.MovieID
            AND Movies.MovieName = "Hacksaw Ridge"
    );
```

10)

```
SELECT CustomerName
FROM Movies, Rentals, Inventory, Customers
WHERE Movies.MovieID = Inventory.MovieID
    AND Rentals.TapeID = Inventory.TapeID
    AND Customers.CustomerID = Rentals.CustomerID
    AND Movies.MovieName = "Beauty and the Beast"
UNION
SELECT CustomerName
FROM Rentals, Inventory, MovieSupplier, Suppliers, Customers
WHERE Rentals.CustomerID = Customers.CustomerID
    AND Inventory.TapeID = Rentals.TapeID
    AND MovieSupplier.SupplierID = Suppliers.SupplierID
    AND Inventory.MovieID = MovieSupplier.MovieID
    AND Inventory.MovieID IN
    (
        SELECT MovieID
        FROM MovieSupplier, Suppliers
        WHERE MovieSupplier.SupplierID = Suppliers.SupplierID
            AND Suppliers.SupplierName = "VWS Video"
    );
```