1. 
```
CREATE TABLE Customers (
  customerNumber INTEGER PRIMARY KEY,
  customerName TEXT NOT NULL,
  contactLastName TEXT NOT NULL,
  contactFirstName TEXT NOT NULL,
  phone TEXT NOT NULL,
  addressLine1 TEXT NOT NULL,
  addressLine2 TEXT NULL,
  city TEXT NOT NULL,
  state TEXT NULL,
  postalCode TEXT NULL,
  country TEXT NOT NULL,
  salesRepEmployeeNumber INTEGER NULL,
  creditLimit REAL NULL
);

CREATE TABLE Employees (
  employeeNumber INTEGER PRIMARY KEY,
  lastName TEXT NOT NULL,
  firstName TEXT NOT NULL,
  extension TEXT NOT NULL,
  email TEXT NOT NULL,
  officeCode TEXT NOT NULL,
  reportsTo INTEGER NULL,
  jobTitle TEXT NOT NULL,
  FOREIGN KEY (reportsTo) REFERENCES Employees(employeeNumber)
);

CREATE TABLE Offices (
  officeCode TEXT PRIMARY KEY,
  city TEXT NOT NULL,
  phone TEXT NOT NULL,
  addressLine1 TEXT NOT NULL,
  addressLine2 TEXT NULL,
  state TEXT NULL,
  country TEXT NOT NULL,
  postalCode TEXT NOT NULL,
  territory TEXT NOT NULL
);

CREATE TABLE OrderDetails (
  orderNumber INTEGER NOT NULL,
  productCode TEXT NOT NULL,
  quantityOrdered INTEGER NOT NULL,
  priceEach REAL NOT NULL,
  orderLineNumber INTEGER NOT NULL,
  PRIMARY KEY (orderNumber, productCode),
  FOREIGN KEY (productCode) REFERENCES Products
);
```

```
CREATE TABLE Orders (
  orderNumber INTEGER PRIMARY KEY,
  orderDate TEXT NOT NULL,
  requiredDate TEXT NOT NULL,
  shippedDate TEXT NULL,
  status TEXT NOT NULL,
  comments TEXT NULL,
  customerNumber INTEGER NOT NULL,
  FOREIGN KEY (customerNumber) REFERENCES Customers
);

CREATE TABLE Payments (
  customerNumber INTEGER NOT NULL,
  checkNumber TEXT NOT NULL,
  paymentDate TEXT NOT NULL,
  amount REAL NOT NULL,
  PRIMARY KEY (customerNumber, checkNumber),
  FOREIGN KEY (customerNumber) REFERENCES Customers
);

CREATE TABLE Products (
  productCode TEXT PRIMARY KEY,
  productName TEXT NOT NULL,
  productLine TEXT NOT NULL,
  productScale TEXT NOT NULL,
  productVendor TEXT NOT NULL,
  productDescription TEXT NOT NULL,
  quantityInStock INTEGER NOT NULL,
  buyPrice REAL NOT NULL,
  MSRP REAL NOT NULL,
  FOREIGN KEY (productLine) REFERENCES Productlines
);

CREATE TABLE ProductLines(
  productLine TEXT PRIMARY KEY,
  description TEXT NULL
);
```

2.

a)
```
SELECT customerName, contactLastName, contactFirstName
FROM   Customers;
```

Output all relations from the three attributes customerName, contactLastName, and contactFirstName from the table Customers. This query gives an overview of the contact persons for each customer company.

b)
```
SELECT *
FROM Orders
WHERE  shippedDate IS NULL;
```

Output all tuples with no shippedDate from the table Orders. This query gives an overview of all orders that have not been reported shipped (although whether or not the order is actually shipped is unknown).

c)
```
SELECT C.customerName AS Customer, SUM(OD.quantityOrdered) AS Total
FROM   Orders O, Customers C, OrderDetails OD
WHERE  O.customerNumber = C.customerNumber
  AND  O.orderNumber = OD.orderNumber
GROUP BY O.customerNumber
ORDER BY Total DESC;
```

Output customer name and total amount of orders, where total amount of orders is computed by taking all orders associated with each customer number, and summing up the quantities from their according order details. The output is in descending order, ordered by the total amount of orders. This shows how many items each customer has ordered, prioritising the ones who have ordered the most items.

d)
```
SELECT P.productName, T.totalQuantityOrdered
FROM   Products P NATURAL JOIN
        (SELECT productCode, SUM(quantityOrdered) AS totalQuantityOrdered
        FROM   OrderDetails GROUP BY productCode) AS T
WHERE  T.totalQuantityOrdered >= 1000;
```

Output the product name of each product along with the total quantity ordered, so long as the total quantity is greater than 1000. The total quantity ordered is computed by adding up the quantity of each individual order made for each product number, then naturally joined with the product table by product number. This gives an overview of how many

3.

1. ```
   SELECT  customerName
   FROM    Customers
   WHERE   UPPER(TRIM(country)) IS 'NORWAY';
   ```
2. ```
   SELECT productName, productScale
   FROM Products
   WHERE productLine IS 'Classic Cars';
   ```
3. ```
   SELECT  O.orderNumber, O.requiredDate, P.productName,
   OD.quantityOrdered, P.quantityInStock
   FROM    OrderDetails OD, Orders O, Products P
   WHERE   OD.orderNumber = O.orderNumber
     AND   OD.productCode = P.productCode
     AND   O.status = 'In Process';
   ```
4. ```
   SELECT  C.customerName, C.creditLimit, Pr.totalPrice, Pm.totalPayment,
   Pr.totalPrice-Pm.totalPayment AS diffPricePayment
   FROM    Customers C NATURAL JOIN
           (
             SELECT  customerNumber, SUM(amount) AS totalPayment
             FROM    Payments
             GROUP BY customerNumber
           ) AS Pm,
           (
             SELECT O.customerNumber, SUM(OD.quantityOrdered*OD.priceEach) AS totalPrice
             FROM Orders O, OrderDetails OD
             WHERE O.orderNumber = OD.orderNumber
             GROUP BY O.customerNumber
           ) AS Pr
   WHERE   C.customerNumber = Pr.customerNumber
     AND   C.customernumber = Pm.customerNumber
     AND   diffPricePayment > C.creditLimit;
   ```
5. ```
   SELECT  customerNumber, customerName
   FROM  (SELECT DISTINCT customerNumber, customerName, productCode
         FROM   Customers C NATURAL JOIN Orders O NATURAL JOIN OrderDetails OD NATURAL JOIN
                 (SELECT  sOD.productCode
                 FROM    Orders sO NATURAL JOIN OrderDetails sOD
                 WHERE   sO.customerNumber = 219) KL/* key list*/) L --actual list
   GROUP BY  customerNumber
   HAVING  COUNT(*) IN (SELECT COUNT(*)
           FROM    Orders sO NATURAL JOIN OrderDetails sOD
           WHERE   sO.customerNumber = 219)
     AND customerNumber != 219
   ORDER BY customerNumber;
   ```
   ```