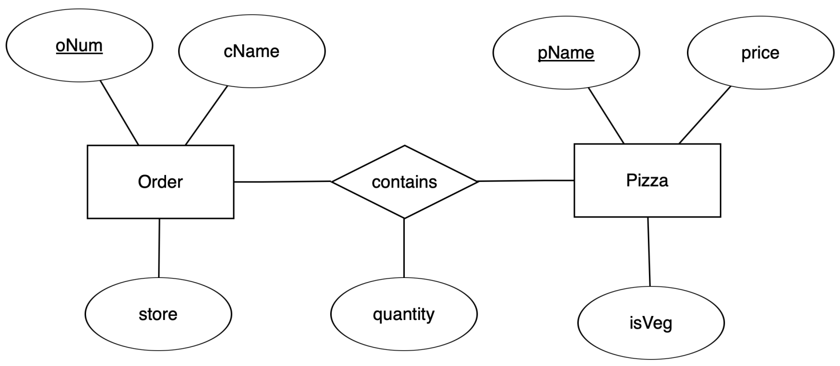
CITS1402 (Semester 2, 2021) — Mid-term Test Schema

Acme Pizza currently uses a database with the following ERD.



There are three tables in Acme's database, with the following schemas:

```
Order(oNum, cName, store)
Pizza(pName, price, isVeg)
contains(oNum, pName, quantity)
```

The meanings of these tables is as follows:

- The table `Order` contains information about an order for pizza. Each order has a unique order number, `oNum`, and is made by a customer called `cName` and delivered from one Acme's stores. A sample row of this table is

  ```
  (100, 'Bill Jones', 'Subiaco')
  ```

  indicating that order number 100 was made by Bill Jones, and delivered from the Subiaco store. There is one row in `Orders` for each order.

- The table `Pizza` contains information about a particular type of pizza. Each pizza has a unique name, `pName` and a `price` (the price that the customer pays) and a field `isVeg` which takes the value 1 if the pizza is suitable for vegetarians and 0 otherwise. There is one row for each type of pizza sold by Acme. Sample rows of this table are

  ```
  ('Margherita', 10.99, 1)
  ('Pepperoni', 13.99, 0)
  ```

  indicating that the Margherita costs \$10.99 and is suitable for vegetarians, but the Pepperoni pizza costs \$13.99 and is definitely *not* suitable for vegetarians. There is one row of `Pizza` for each type of pizza sold by Acme.

- The table `contains` indicates which pizzas, and how many of them, are contained in each order. The columns `oNum` and `pName` refer to the order number and type of pizza respectively, while `quantity` indicates how many of this type pizza are in this order. Some sample rows of this table are

  ```
  (100, 'Margherita', 2)
  (100, 'Pepperoni', 1)
  ```

  indicating that order 100 contains 2 Margherita and 1 Pepperoni pizza. For a single order, there will be one row of `contains` for each type of pizza in that order.

1. Write a suitable SQLite DDL statement that will *create* a table called `Pizza`, as described above. Use the most appropriate SQLite data types for the columns, and do not attempt to use foreign keys. In addition, give a DDL statement to *insert* two rows into this table, using the details of the Margherita and Pepperoni pizzas as given above.

```
CREATE TABLE Pizza (
  pName TEXT PRIMARY KEY,
  price REAL,
  isVeg INTEGER);

INSERT INTO Pizza VALUES ('Margherita',10.99,1), ('Pepperoni',13.99,0)
```

2. Write a single SQL query that lists the *name* and *price* of each of the pizzas, in increasing order of price.

```
SELECT name, price
FROM Pizza
ORDER BY price;
```

3. Write a single SQL query that lists the *name* of each customer, and the *number of orders* made by that customer. Ensure that the output columns are called `customerName` and `numberOrders`.

```sql
SELECT cName AS customerName, COUNT(*) AS numberOrders
FROM Order
GROUP BY cName;
```

4. Write a single SQL query that, for each order, lists the *order number* and the *total price* of the order; the total price of an order is the cost of the pizzas in the order, plus a flat $5 delivery fee.

```
SELECT oNum, 5 + SUM(quantity*price)
FROM contains JOIN Pizza USING (pName)
GROUP BY oNum;
```

5. Acme Pizza decides to start a loyalty card scheme. It will issue a *bronze* card to all customers who have made fewer than 10 orders and a *silver* card to those who have made at least 10 orders. Write a single SQL query that lists the name of each customer, and then the word `'bronze'` or `'silver'` according to the card they should receive.

```sql
SELECT cName,
CASE WHEN COUNT(*) >= 10 THEN 'silver'
     ELSE 'bronze' END
FROM Order
GROUP BY cName;
```

```sql
SELECT cName, IIF (COUNT(*) >= 10, 'silver', 'bronze')
FROM Order
GROUP BY cName;
```

6. Write a single SQL query, using an *uncorrelated subquery*, that lists the *names*, without duplicates, of customers who have only ever ordered vegetarian pizzas.

```
SELECT DISTINCT cName
FROM Order
WHERE cName
NOT IN (SELECT cName
        FROM Order JOIN contains USING (oNum)
                   JOIN Pizza USING (pName)
                   WHERE isVeg = 0);
```