```
from threading import Thread
from queue import Queue

q = Queue()
final_results = []

def producer():
    for i in range(100):
        q.put(i)

def consumer():
    while True:
        number = q.get()
        result = (number, number**2)
        final_results.append(result)
        q.task_done()


# Create 5 different threads that each runs the consumer function
for i in range(5):
    t = Thread(target=consumer)
    t.daemon = True
    t.start()

# Start creating the items and adding them to the queue
producer()

# Wait until queue is empty and task_done() has been called on each element
q.join()

print (final_results)
```

Answer the following questions:

1. **How is the queue data structure used to achieve the purpose of the code?**

   The queue allows the elements created by the producer to be stored in a predictable order and as it is a FIFO data structure, it allows the elements to be accessed by the consumer in the same order they were added.

2. **What is the purpose of q.put(I)?**

   This adds the integers in the 0 - 100 loop to the Queue q.

3. **What is achieved by q.get()?**

   This fetches the next item from the head of the q.

4. **What functionality is provided by q.join()?**
   It forces the program to wait until the queue is empty and task_done(0 has been called on each element

5. **Extend this producer-consumer code to make the producer-consumer scenario available in a secure way. What technique(s) would be appropriate to apply?**

   Some potential issues I can think of would be the Queue filling up faster than the consumer empties it, and potentially leading to a ‘buffer overflow’ error when the Queue is full.
   In Python, you can make the size of the queue unlimited by passing in the size=0 parameter when constructing the queue.

   Of course, I don’t think that a queue actually has infinite space, so it would be worthwhile checking if the queue has capacity before writing a new value to it.

   There is a good explanation about how the Queue data structure is used in a muti-threading context here:
   https://www.troyfawkes.com/learn-python-multithreading-queues-basics/