EXERCISE 2

Q1: Can you explain the purpose of the orders_items table?

Answer: The main purpose of the orders_items is to have track of the every item that we have ordered. This table contains major details like id of the items that are ordered, id of the order (which typically holds the order details like when it is ordered, unique id to track the order etc.), id of the product (this typically holds the product details). Thus this is a bridge table between the order table and the product table, which allows us to link each order to the products that were purchased in that order.

Q2: Can you write a SQL query to find the average order cost per country?

Answer: The below query joins the customers and orders tables on the customer ID, groups the results by country, and calculates the average order cost for each country.

```
SELECT cust.country, AVG(ord.order_cost) as avg_order_cost_cty
FROM customers cust
JOIN orders ord ON cust.id = ord.id_customer
GROUP BY cust.country;
```

Q3: Can you write a SQL query to find the name of the highest price product sold to an Italian customer?

Answer: The below query joins the customers, orders, orders_items, and products tables, which filters the results to only include orders made by Italian customers. This sorts the products by price in descending order, and returns the name of the most expensive product sold to an Italian customer.

```
SELECT prod.name
FROM customers cust
JOIN orders ord ON cust.id = ord.id_customer
JOIN orders_items ord_item ON ord.id = ord_item.id_order
JOIN products prod ON ord_item.id_product = prod.id
WHERE cust.country = 'Italy'
ORDER BY prod.price DESC
LIMIT 1;
```

Q4: How could you optimize the orders table assuming you have to manage a lot of queries?

Answer: For optimizing the orders table to manage loads of queries, we can use the indexing approach by adding indexes to the table. Indexes allow the database in quick search and retrieval of specific rows based on the values in certain columns. In this case, we should essentially need to add an index to the id_customer column, since many queries will likely involve joining the orders table with the customers table on that column.

Q5: What would you change if the amount of data is too big to run these queries? (suppose to have 500TB of data)

Answer: If the amount of data is huge to run the queries efficiently, we can follow the method of chunking the data into smaller parts/chunks/pieces which can be typically managed to use. Also using a distributed database system will help in distributing the data across multiple servers which in turn thrives to improve the performance and efficacy. Additionally, usage of caching techniques to store frequently accessed data in memory and it reduces the number of disk reads required.