Generating insights from data

This assignment is about extrating data from a database with SQL and exploring different ways of generating additional data insights through advanced SQL statement.

You are working as a UX Designer at a small web agency. One of the large clients wants to implement some changes to their e-commerce solution and you have just been assigned the team working on it. The backend developer, Sheila, informs you that she is very busy right working on the database but there are a few tasks you can start on right now. Since you have little experience with databases she asks you to just jump in and try to figure it out.

For these tasks Sheila has provided you with a dump of the database: **dump.sql**

1) Customer names from orders

The client wants to have a list of all customers that have made an order so that they can import it into their CRM system.

Sheila has provided you with a SQL snippet that she is using the get all the order ids and customer names from the orders table.

```
SELECT orders.id, customers.firstName, customers.lastName
FROM orders
INNER JOIN customers  ON orders.customer_id = customers.id
```

Modify it as follows:

1) Skip orders id, we don't need it
2) Make it only show each customer name once (hint: she told you to use "group by")
3) Get it to include the customer email

2) Customer without orders

The client wants to send an email to all customers who haven't made an order yet, so their marketing department requested to get a list of emails and name.

The backend developer told you:
"Use a WHERE clause to limit the query. Remember that it returns NULL for those without a match".

You think you might know what she is talking about. If you do a LEFT JOIN you will get all the customers plus what ever matches in the orders table. If nothing matches, order id would be NULL. So you try:

```
SELECT customers.firstName, customers.lastName, orders.id
FROM customers
LEFT JOIN orders  ON customers.id = orders.customer_id
```

firstName	lastName	id
Johnny	Hanson	1
Johnny	Hanson	3
Silvia	Banderlust	2
Frank	Zappa	*NULL*
Marie	Pencil	*NULL*

Sure enough, the customers without any orders return as order id NULL.

1) Make sure the customer email is included
2) Limit the query to only customers where order id is null

3) Count the countries the customers are from

The client want to know how many customers are coming from each country.

Sheila told you that it is possible to count the occurrences of rows with the same value using the COUNT() method. She also talks about returning this as a auto-generated row by using AS.

She send you this example from another project:

```
Select COUNT(*) AS countBooks, authors
From library
Group By authors
```

She described it as follows:
"Firstly, use the COUNT() method to count the number of rows and give it a name with the AS assignment. Then make sure you group by so that you won't get multiple records for each country."

4) Who bought the most products (optional)

The client want to know which customers bought the highest quantity of which products.

Sheila said this was tricky but could properly be done by using multiple joins and the COUNT() method. Maybe ask Thommy, she said. Thommy is working with app development as Senior Developer and he gave you the following snippet, saying "just do something like this".

```
SELECT COUNT(*) AS qty, customers.firstName
FROM customers

JOIN orders
ON orders.customer_id = customers.id

JOIN products_in_orders
ON products_in_orders.order_id = orders.id
```

```
JOIN products
ON products.id = products_in_orders.product_id

GROUP BY products.name, customers.email
```

1) Make sure you get all the relevant customer info (name, email)
2) Also get the product name and description
3) Order the list so that it starts with the highest amount (hint: ASC gives you ascending numbers and DESC descending)

5) Most valuable customer (optional)

Having the information about who bought the most, the client now want to know who bought for the most money.

Thommy gives you the following advise: "In your select statement you can create an additional row where you simply take the COUNT() and multiply it with the product price. Assign it the name "value" and voila!"

1) Build on top of the previous SQL query, add another column in your select statement where you use the same count method and call it AS value.
2) Multiply the count with the product price
3) Sort DESC on value