GROUP 17 - Online Vinyl Record Store - Daria Solomon, Timeea Andreea Radu, Ioana Lupu

Select Statements:

1. customers who have not purchased any vinyls in the last 6 months
   This one has one JOIN, no aggregation function and no GROUP BY

```
MariaDB [sdaria_db]> SELECT c.CustomerID, c.Name
    -> FROM Customer c
    -> LEFT JOIN Payments p ON c.CustomerID = p.CustomerID AND p.PaymentFor = 'Vinyl Purchase'
    -> WHERE p.PaymentDate < NOW() - INTERVAL 6 MONTH OR p.PaymentID IS NULL;
+------------+--------------+
| CustomerID | Name         |
+------------+--------------+
|          4 | Diana Carter |
+------------+--------------+
1 row in set (0.001 sec)
```

2. names and total purchases of customers who bought vinyls in a specific genre: This one has 2 joins, one aggregation function (COUNT) and one GROUP BY

```
MariaDB [sdaria_db]> SELECT c.CustomerID, c.Name, COUNT(p.PaymentID) AS VinylPurchases
    -> FROM Payments p
    -> JOIN Vinyl v ON p.VinylID = v.VinylID
    -> JOIN Customer c ON c.CustomerID = p.CustomerID
    -> WHERE v.Genre = 'Rock'
    -> GROUP BY c.CustomerID;
+------------+---------------+----------------+
| CustomerID | Name          | VinylPurchases |
+------------+---------------+----------------+
|          1 | Alice Johnson |              1 |
|          2 | Bob Smith     |              2 |
|          3 | Charlie Davis |              1 |
+------------+---------------+----------------+
3 rows in set (0.001 sec)
```

3. most expensive vinyl purchased by each customer along with the price
   This includes 2 joins, one aggregation function (MAX) and GROUP BY

```
MariaDB [sdaria_db]> SELECT c.CustomerID, c.Name, v.Name AS VinylName, MAX(v.Price) AS MaxPrice
    -> FROM Payments p
    -> JOIN Vinyl v ON p.VinylID = v.VinylID
    -> JOIN Customer c ON c.CustomerID = p.CustomerID
    -> WHERE p.PaymentFor = 'Vinyl Purchase'
    -> GROUP BY c.CustomerID;
+------------+---------------+--------------+----------+
| CustomerID | Name          | VinylName    | MaxPrice |
+------------+---------------+--------------+----------+
|          1 | Alice Johnson | Abbey Road   |    29.99 |
|          2 | Bob Smith     | Back in Black |    34.99 |
|          3 | Charlie Davis | Thriller     |    39.99 |
+------------+---------------+--------------+----------+
```

4. Returns the number of albums that are available in the shop by a certain artist. We display the amount in decreasing order. We use ORDER BY, GROUP BY and aggregate function COUNT.

```
MariaDB [rtimeea_andreea_db]> SELECT DISTINCT v.Artist, COUNT(v.Name)AS Vinyl_nb FROM Vinyl v
GROUP BY v.Artist ORDER BY Vinyl_nb DESC;
+-----------------+----------+
| Artist          | Vinyl_nb |
+-----------------+----------+
| Michael Jackson |        2 |
| AC/DC           |        1 |
| The Beatles     |        1 |
| Pink Floyd      |        1 |
+-----------------+----------+
4 rows in set (0.001 sec)
```

5. Returns how many purchases a regular customer has. It is useful if we would want to advise them to buy a membership after exceeding a purchase amount or maybe price amount(using HAVING). We use GROUP BY and one aggregation function COUNT().

```
MariaDB [rtimeea_andreea_db]> SELECT p.CustomerID, COUNT(p.PaymentID) AS Payment
s_done,p.PaymentFor
    -> FROM Payments p
    -> WHERE p.DiscountID IS NULL AND p.PaymentFor = "Vinyl Purchase"
    -> GROUP BY CustomerID;
+------------+---------------+----------------+
| CustomerID | Payments_done | PaymentFor     |
+------------+---------------+----------------+
|          4 |             2 | Vinyl Purchase |
+------------+---------------+----------------+
1 row in set (0.001 sec)

MariaDB [rtimeea_andreea_db]>
```

```
SELECT p.CustomerID, COUNT(p.PaymentID),p.PaymentFor
FROM Payments p
WHERE p.DiscountID IS NULL AND p.PaymentFor = "Vinyl Purchase"
GROUP BY CustomerID
HAVING COUNT(p.PaymentID) > 5;  -- add this with more data in order to suggest the customer to buy a membership
```

6. Returns a table which shows the artists that were purchased the most. We use the aggregate function COUNT, a LEFT JOIN and a GROUP BY.

```
MariaDB [rtimeea_andreea_db]> SELECT p.VinylID, COUNT(p.VinylID) AS Vinyl_nb, v.Artist
    -> FROM Payments p
    -> LEFT JOIN Vinyl v ON p.VinylID = v.VinylID
    -> WHERE p.VinylID IS NOT NULL
    -> GROUP BY p.VinylID;
+---------+----------+----------------+
| VinylID | Vinyl_nb | Artist         |
+---------+----------+----------------+
|       1 |        2 | The Beatles    |
|       2 |        1 | Pink Floyd     |
|       3 |        1 | Michael Jackson |
|       4 |        1 | AC/DC          |
|       5 |        2 | Michael Jackson |
+---------+----------+----------------+
5 rows in set (0.001 sec)

MariaDB [rtimeea_andreea_db]>
```

7. Returns the most expensive vinyl from every genre. Here we have the aggregation function (Max) and a GROUP BY.

```
MariaDB [lioana_db]> SELECT v.Genre, Max(v.Price) as MostExpensiveVinyl
    -> FROM Vinyl v
    -> WHERE v.Availability > 0
    -> GROUP BY v.Genre;
+-------+--------------------+
| Genre | MostExpensiveVinyl |
+-------+--------------------+
| Pop   |              44.99 |
| Rock  |              34.99 |
+-------+--------------------+
```

8. Returns the average price of vinyls purchased in the last 3 months. Here we have one left join and the aggregation function (AVG).

```
MariaDB [lioana_db]> SELECT AVG(v.Price) AS AvgPriceVinyls
    ->  FROM Vinyl v
    ->  LEFT JOIN Payments p ON v.VinylID = p.VinylID
    ->  WHERE p.PaymentFor = 'Vinyl Purchase' AND  p.PaymentDate >= NOW() - INTERVAL 3 MONTH;
+----------------+
| AvgPriceVinyls |
+----------------+
|      31.990000 |
+----------------+
```

9. Returns how many vinyls of each type (EP,LP,Single) there are for each artist. Here we have three left joins, the aggregation function(COUNT) and a GROUP BY.

```
MariaDB [lioana_db]> SELECT v.Artist,
    -> COUNT(e.VinylID) AS Num_EP, COUNT(l.VinylID) AS Num_LP, COUNT(s.VinylID) AS Num_Single
    -> FROM Vinyl v
    -> LEFT JOIN EP e ON v.VinylID = e.VinylID
    -> LEFT JOIN LP l on v.VinylID = l.VinylID
    -> LEFT JOIN Single s on v.VinylID = s.VinylID
    -> GROUP BY v.Artist;
+-----------------+--------+--------+------------+
| Artist          | Num_EP | Num_LP | Num_Single |
+-----------------+--------+--------+------------+
| AC/DC           |      0 |      0 |          1 |
| Michael Jackson |      1 |      0 |          0 |
| Pink Floyd      |      0 |      1 |          0 |
| The Beatles     |      0 |      1 |          0 |
+-----------------+--------+--------+------------+
4 rows in set (0.001 sec)
```