SQL Extraction Code

```
-- This SQL query is designed to aggregate user data for A/B testing
analysis.
-- The goal is to join the 'users', 'groups', and 'activity' tables
-- to obtain a comprehensive dataset that includes user
--demographics,
-- their group assignment (test/control), and their activity data
--(amount spent).
SELECT
  -- Select relevant columns from the 'users' table
  u1.id as users_id, -- User ID
  u1.country as users_country, -- Country of the User
  u1.gender as users_gender, -- Gender of the User
  -- Select relevant columns from the 'groups' table
  g1.device as users_device, -- Device used by the User
  g1.group as test_group, -- Indicates if the user is in the test or control
group
  -- Calculate the 'Converted' column:
  -- If the total amount spent is greater than 0, mark it as 'Yes',
otherwise 'No'
```

```
  CASE
    WHEN SUM(a1.spent) > 0 THEN 'Yes'
    ELSE 'No'
  END AS Converted,
  -- Calculate 'total_spent' as the sum of the 'spent' column from
--'activity'
  -- If there is no data (NULL), replace it with 0
  COALESCE(SUM(a1.spent), 0) AS total_spent
-- Joining Tables
FROM users u1 -- Start with the 'users' table
-- Left join with 'groups' on user ID
LEFT JOIN groups g1
ON u1.id = g1.uid
-- Left join with 'activity' on user ID
LEFT JOIN activity a1
ON u1.id = a1.uid
-- Group the data by user ID, country, gender, device type, and
--group type
GROUP BY u1.id,
        u1.country,

        u1.gender,

        g1.device,

        G1.group

-- Sort the results by user ID in ascending order

ORDER BY u1.id ASC;
```

Code For TheNovelty Effect

```
-- Start the SQL query by selecting the columns we need
SELECT
   a1.dt,                          -- Date of the activity
   g1.group as test_group,             -- Group to which the user belongs
  COUNT(DISTINCT a1.uid) as total_users,  -- Count of unique
users
   SUM(ROUND(a1.spent,2)) as total_purchase, -- Sum of all
purchases, rounded to 2 decimal places
   AVG(ROUND(a1.spent,2)) as average_purchase  -- age purchase
per us  Averer, rounded to 2 decimal places
-- From the 'activity' table, aliased as 'a1'
FROM
   activity a1
```

```sql
-- Perform a JOIN operation with the 'groups' table, aliased as 'g1'
-- Matching is done based on the 'uid' column in both tables
JOIN
   groups g1
   ON a1.uid = g1.uid
-- Filter the rows based on the date range, from '2023-01-25' to
'2023-02-06'
WHERE
   a1.dt BETWEEN  '2023-01-25' AND '2023-02-06'
-- Group the result set by date and group
GROUP BY
   a1.dt,
   g1.group

-- Sort the result set first by date in descending order, then by group
ORDER BY
   a1.dt DESC,
   g1.group;
```