GAME ANALYTICS

delivery 1 - KPIs

Antonio Romanos, Joel Maldonado, Víctor Gil, Daniel Mariages

Objectives

Data / Patterns / Hidden trends

- The objectives are:

  - Collect data

  - To analyse them

  - Discover trends and patterns

  For this, we will collect player data such as age, gender or number of connections, among others.

Gather & store

Client Code

```
// Coroutine to upload player data to the server
IEnumerator UploadPlayer(string name, string country, int age, float gender, DateTime date)
{
    // Create a form with the necessary fields
    WWWForm form = new WWWForm();
    form.AddField("name", name);
    form.AddField("country", country);
    form.AddField("age", age.ToString());
    form.AddField("gender", gender.ToString(System.Globalization.CultureInfo.InvariantCulture));
    form.AddField("date", date.ToString("yyyy-MM-dd HH:mm:ss"));

    // Send form data to the server
    using (UnityWebRequest www = UnityWebRequest.Post("https://citmalumnes.upc.es/~antoniorr14/Player_Data.php", form))
    {
        yield return www.SendWebRequest();

        // Check for success or error in the response
        if (www.result != UnityWebRequest.Result.Success)
        {
            UnityEngine.Debug.LogError("Player data upload failed: " + www.error);
        }
        else
        {
            // Parse the response text to retrieve the player ID
            string answer = www.downloadHandler.text.Trim(new char[] { '\uFEFF', '\u200B', ' ', '\t', '\r', '\n' });
            if (uint.TryParse(answer, out uint parsedId) && parsedId > 0)
            {
                currentUserId = parsedId;  // Set the current user ID
                CallbackEvents.OnAddPlayerCallback.Invoke(currentUserId);  // Trigger callback with new player ID
                Debug.Log("Uploaded Player with ID: " + currentUserId);
            }
            else
            {
                UnityEngine.Debug.LogError("Invalid user ID received: " + answer);
            }
        }
    }
}
```

```
// Coroutine to upload session start data to the server
IEnumerator UploadStartSession(DateTime date, uint playerID)
{
    // Check if currentUserId is valid before starting session
    if (currentUserId == 0)
    {
        UnityEngine.Debug.LogError("User ID is not set, cannot start session");
        yield break;  // Exit coroutine if UserId is not valid
    }

    // Create a form with the necessary fields
    WWWForm form = new WWWForm();
    form.AddField("UserId", currentUserId.ToString());
    form.AddField("StartSession", date.ToString("yyyy-MM-dd HH:mm:ss"));

    // URL to post the data to
    string url = "https://citmalumnes.upc.es/~antoniorr14/NewSession.php";
    UnityWebRequest www = UnityWebRequest.Post(url, form);
    yield return www.SendWebRequest();

    // Check for success or error in the response
    if (www.result == UnityWebRequest.Result.Success)
    {
        // Parse the response text to retrieve the session ID
        string answer = www.downloadHandler.text.Trim(new char[] { '\uFEFF', '\u200B', ' ', '\t', '\r', '\n' });
        if (uint.TryParse(answer, out uint parsedId) && parsedId > 0)
        {
            currentSessionId = parsedId;  // Set the current session ID
            CallbackEvents.OnNewSessionCallback.Invoke(currentSessionId);  // Trigger callback with new session ID
        }
        else
        {
            UnityEngine.Debug.LogError("Invalid session ID received: " + answer);
        }
    }
    else
    {
        UnityEngine.Debug.LogError("Session start data upload failed: " + www.error);
    }
}
```

Gather & store

Client Code

```
// Coroutine to upload purchase data to the server
IEnumerator UploadItem(int item, DateTime date, uint playerID)
{
    // Create a form with the necessary fields
    WWWForm form = new WWWForm();
    form.AddField("Item", item.ToString());
    form.AddField("Session_ID", currentSessionId.ToString());
    form.AddField("Buy_Date", date.ToString("yyyy-MM-dd HH:mm:ss"));

    // URL to post the data to
    UnityWebRequest www = UnityWebRequest.Post("https://citmalumnes.upc.es/~antoniorr14/Purchase_Data.php", form);
    yield return www.SendWebRequest();

    // Check for success or error in the response
    if (www.result != UnityWebRequest.Result.Success)
    {
        UnityEngine.Debug.LogError("Purchase data upload failed: " + www.error);
    }
    else
    {
        // Parse the response text to retrieve the purchase ID
        string answer = www.downloadHandler.text.Trim(new char[] { '\uFEFF', '\u200B', ' ', '\t', '\r', '\n' });
        if (uint.TryParse(answer, out uint parsedId) && parsedId > 0)
        {
            currentPurchaseId = parsedId;  // Set the current purchase ID
            CallbackEvents.OnItemBuyCallback.Invoke(playerID);  // Trigger callback for item purchase
        }
        else
        {
            UnityEngine.Debug.LogError("Invalid purchase ID received: " + www.downloadHandler.text + "ID: " + parsedId);
        }
    }
}
```

```
// Coroutine to upload session end data to the server
IEnumerator UploadEndSession(DateTime date, uint playerID)
{
    // Create a form with the necessary fields
    WWWForm form = new WWWForm();
    form.AddField("User_Id", currentUserId.ToString());
    form.AddField("End_Session", date.ToString("yyyy-MM-dd HH:mm:ss"));
    form.AddField("Session_ID", currentSessionId.ToString());

    // URL to post the data to
    string url = "https://citmalumnes.upc.es/~antoniorr14/CloseSession.php";
    UnityWebRequest www = UnityWebRequest.Post(url, form);
    yield return www.SendWebRequest();

    // Check for success or error in the response
    if (www.result == UnityWebRequest.Result.Success)
    {
        CallbackEvents.OnEndSessionCallback.Invoke(currentSessionId);  // Trigger callback for session end
    }
    else
    {
        UnityEngine.Debug.LogError("Session end data upload failed: " + www.error);
    }
}
```

Gather & store

Server Code

```php
<?php

// Set up database connection parameters
$servername = "localhost";
$username = "antoniorr14";
$password = "46949721m";
$database = "antoniorr14";

// Establish a new MySQL database connection
$conn = new mysqli($servername, $username, $password, $database);

// Verificar conexión
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error); // End script if connection fails
}

// Receive data sent from Unity via POST method
$name = isset($_POST['name']) ? $_POST['name'] : "";              // User's name
$country = isset($_POST['country']) ? $_POST['country'] : "";     // User's country
$age = isset($_POST['age']) ? intval($_POST['age']) : 0;          // User's age (converted to integer)
$gender = isset($_POST['gender']) ? $_POST['gender'] : "";        // User's gender
$install_date = isset($_POST['date']) ? $_POST['date'] : "";      // User's installation date

// Check that all required data fields are provided
if (!empty($name) && !empty($country) && $age > 0 && !empty($gender)) {

    // Prepare a SQL statement to insert data into the UsersInfo table
    $stmt = $conn->prepare("INSERT INTO UsersInfo (`Name`, `Country`, `Age`, `Gender`, `Install_Date`) VALUES (?, ?, ?, ?, ?)");

    // Bind the parameters to the SQL statement (s for string, i for integer)
    $stmt->bind_param("ssiss", $name, $country, $age, $gender, $install_date);

    // Execute the prepared statement and check for success
    if ($stmt->execute()) {
        // If successful, return the ID of the last inserted row
        echo $conn->insert_id;
    } else {
        echo "ERROR: " . $stmt->error;
    }
    // Close the prepared statement
    $stmt->close();

} else {
    // Return an error message if required data is missing
    echo "Missing parameters";
}

// Close the database connection
$conn->close();

?>
```

Player_Data.php

```php
<?php

// Database connection details
$servername = "localhost";
$username = "antoniorr14";
$password = "46949721m";
$database = "antoniorr14";

// Create and verify the connection
$conn = new mysqli($servername, $username, $password, $database);

// Check for connection errors
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Receive data sent from Unity via POST
$userid = isset($_POST['UserId']) ? intval($_POST['UserId']) : 0;
$starttime = isset($_POST['StartSession']) ? $_POST['StartSession'] : "";

// Check if both userid and starttime are provided
if (!empty($userid) && !empty($starttime)) {
    // Use a prepared statement to insert data securely
    $stmt = $conn->prepare("INSERT INTO SessionsData (UserId, StartSession) VALUES (?, ?)");
    $stmt->bind_param("is", $userid, $starttime);

    // Execute the statement and check for success
    if ($stmt->execute()) {
        $last_id = $stmt->insert_id;  // Retrieve the last inserted ID
        echo $last_id;
    } else {
        echo "ERROR: " . $stmt->error;
    }

    // Close the prepared statement
    $stmt->close();
} else {
    echo "Missing parameters";  // Notify if data is missing
}

// Close the database connection
$conn->close();

?>
```

NewSession.php

Gather & store

Server Code

```php
<?php

// Database connection details
$servername = "localhost";
$username = "antoniorr14";
$password = "46949721m";
$database = "antoniorr14";

// Create and verify the connection
$conn = new mysqli($servername, $username, $password, $database);

// Check for connection errors
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Receive POST data for session ending
$sessionId = $_POST["Session_ID"];
$endSession = $_POST["End_Session"];

// Log received data for debugging purposes
error_log("Received end session data: Session_ID={$sessionId}, End_Session={$endSession}");

// Prepare SQL statement to update the session's end time
$stmt = $conn->prepare("UPDATE SessionsData SET `EndSession` = ? WHERE `id` = ?");
$stmt->bind_param("si", $endSession, $sessionId);

// Execute the update and check if any row was affected
if ($stmt->execute()) {
    if ($stmt->affected_rows > 0) {
        // Return the end session time if successful
        echo $endSession;
    } else {
        // Log if no session was updated
        error_log("No session updated in Close_Session_Data.php");
        echo "No session updated";
    }
} else {
    // Log error if update fails
    error_log("Error in Close_Session_Data.php: " . $stmt->error);
    echo "Error: " . $stmt->error;
}

// Close the statement and connection
$stmt->close();
$conn->close();
?>
```

EndSession.php

```php
<?php

// Database connection details
$servername = "localhost";
$username = "antoniorr14";
$password = "46949721m";
$database = "antoniorr14";

// Create and verify the connection
$conn = new mysqli($servername, $username, $password, $database);

// Check for connection errors
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Receive POST data for purchase
$itemId = isset($_POST["Item"]) ? $_POST['Item'] : 0;
$sessionId = isset($_POST["Session_ID"]) ? $_POST['Session_ID'] : 0;
$buyDate = isset($_POST['Buy_Date']) ? $_POST['Buy_Date'] : "";

// Log received data for debugging
error_log("Received purchase data: Session_ID={$sessionId}, Item={$itemId}, Buy_Date={$buyDate}");

// Check if item ID, session ID, and buy date are provided
if(!empty($itemId) && !empty($sessionId) && !empty($buyDate))
{
    // Use a prepared statement to insert purchase data securely
    $stmt = $conn->prepare("INSERT INTO `Purchases`(`itemId`, `buyDate`, `sessionId`) VALUES (?, ?, ?)");
    $stmt->bind_param("isi", $itemId, $buyDate, $sessionId);

    // Execute the statement and check for success
    if ($stmt->execute()) {
        echo $conn->insert_id;  // Output the last inserted ID
    } else {
        // Log error if insertion fails
        error_log("Error in Purchase_Data.php: " . $stmt->error);
        echo "Error: " . $stmt->error;
    }
} else {
    // Notify if data is missing
    echo "Missing parameters";
}

// Close the statement and connection
$stmt->close();
$conn->close();
?>

```

Purchase_Data.php

Gather & store

Database

ID	Name	Country	Age	Gender	Install_Date
1	Vorosethra	Montenegro	22	0.669648	2022-06-12 15:11:34
2	Vorithra	Botswana	87	0.673115	2022-11-10 03:46:09
3	Zubesekira	Congo	48	0.384777	2022-08-27 22:33:59
4	Islyvea	Kiribati	91	0.484933	2022-10-22 12:01:18
5	Gelenn	Kiribati	79	0.106255	2022-10-17 23:21:26
6	Islyldamina	Czech_Republic	76	0.112058	2022-01-14 12:43:32

UsersInfo

Total Users: 502

ID	Name	Country	Age	Gender	Install_Date
501	Vorinn	Malaysia	48	0.754979	2022-07-29 17:11:17
502	Salebrevea	Malaysia	59	0.75212	2022-08-25 23:42:49

Id	Price
1	0.99
2	1.99
3	9.99
4	49.99
5	99.99

Items

purchaseID	itemId	buyDate	sessionId
1	1	2022-02-02 21:32:43	17
2	3	2022-02-03 05:46:15	18
3	2	2022-06-03 07:43:07	30
4	1	2022-06-03 12:46:54	32
5	2	2022-06-04 18:33:06	38
6	1	2022-05-01 11:18:51	49
7	2	2022-05-01 19:25:46	51
8	3	2022-06-27 03:38:22	56
9	1	2022-06-29 13:33:44	67
10	5	2022-06-30 18:07:04	73

Purchases

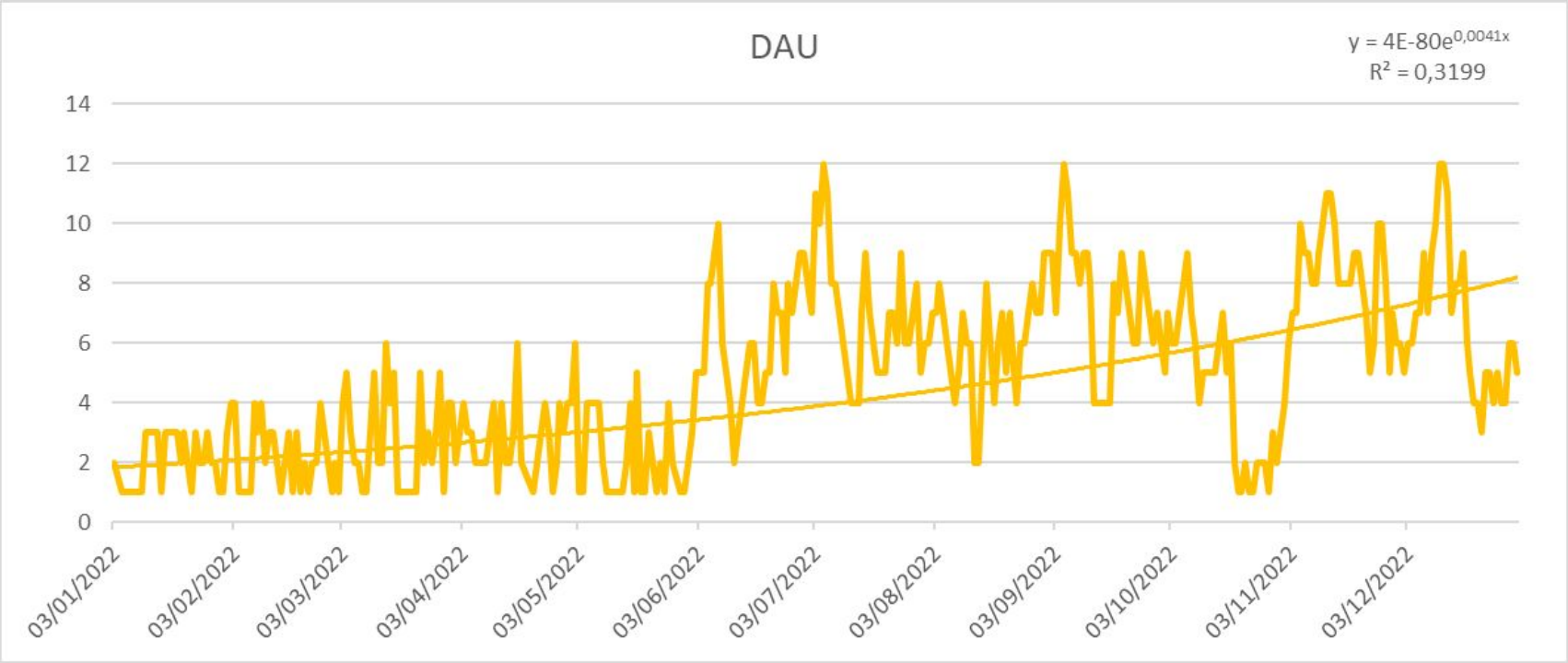
id	UserId	StartSession	EndSession
1	3	2022-08-27 22:33:59	2022-08-27 22:46:21
2	3	2022-08-28 03:34:56	2022-08-28 03:47:01
3	3	2022-08-28 10:08:18	2022-08-28 10:13:43
4	3	2022-08-28 15:38:39	2022-08-28 15:45:13
5	3	2022-08-28 18:59:16	2022-08-28 19:12:11
6	3	2022-08-28 23:35:50	2022-08-28 23:39:09
7	3	2022-08-29 06:25:59	2022-08-29 06:32:37
8	3	2022-08-29 11:44:35	2022-08-29 11:52:45

SessionsData

Visualize & Report

DAU

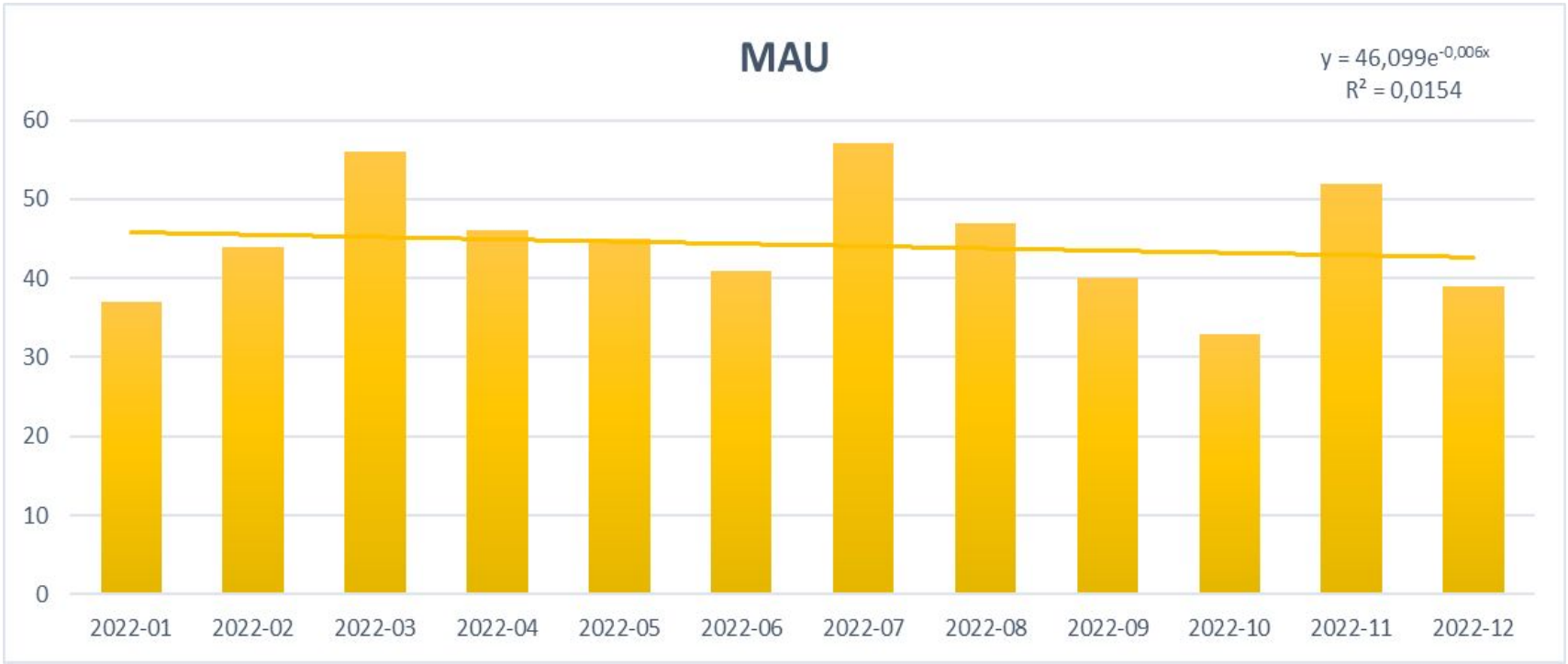
```sql
SELECT
    DATE(StartSession) AS Day,
    COUNT(DISTINCT UserId) AS DAU
FROM
    SessionsData
GROUP BY
    Day
ORDER BY
    Day;
```



Visualize & Report

MAU

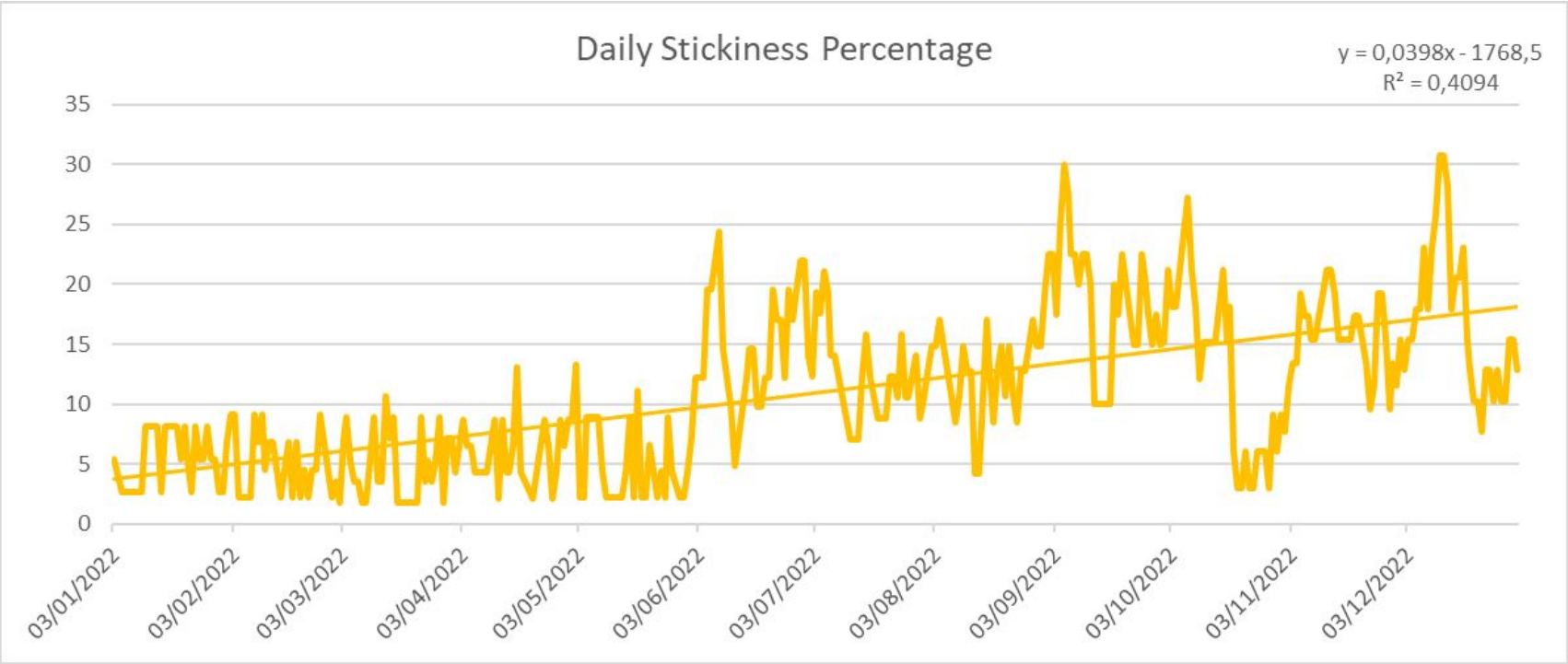
```sql
SELECT
    DATE_FORMAT(StartSession, '%Y-%m') AS Month,
    COUNT(DISTINCT UserId) AS MAU
FROM
    SessionsData
GROUP BY
    Month
ORDER BY
    Month;
```



Visualize & Report

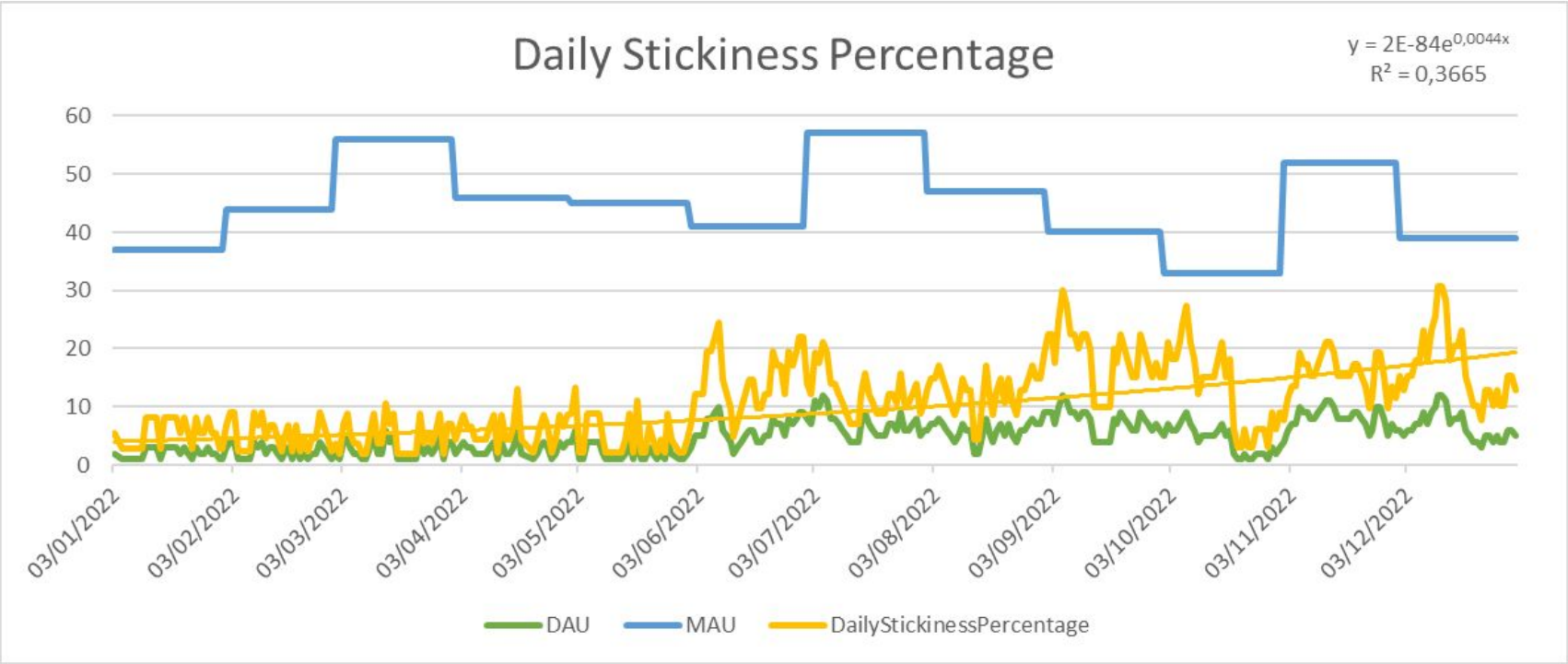
Stickiness

```sql
SELECT
    DATE_FORMAT(StartSession, '%Y-%m') AS Month,
    (COUNT(DISTINCT DATE(StartSession), UserId) / COUNT(DISTINCT UserId)) AS Stickiness
FROM
    SessionsData
GROUP BY
    Month
ORDER BY
    Month;
```



Visualize & Report

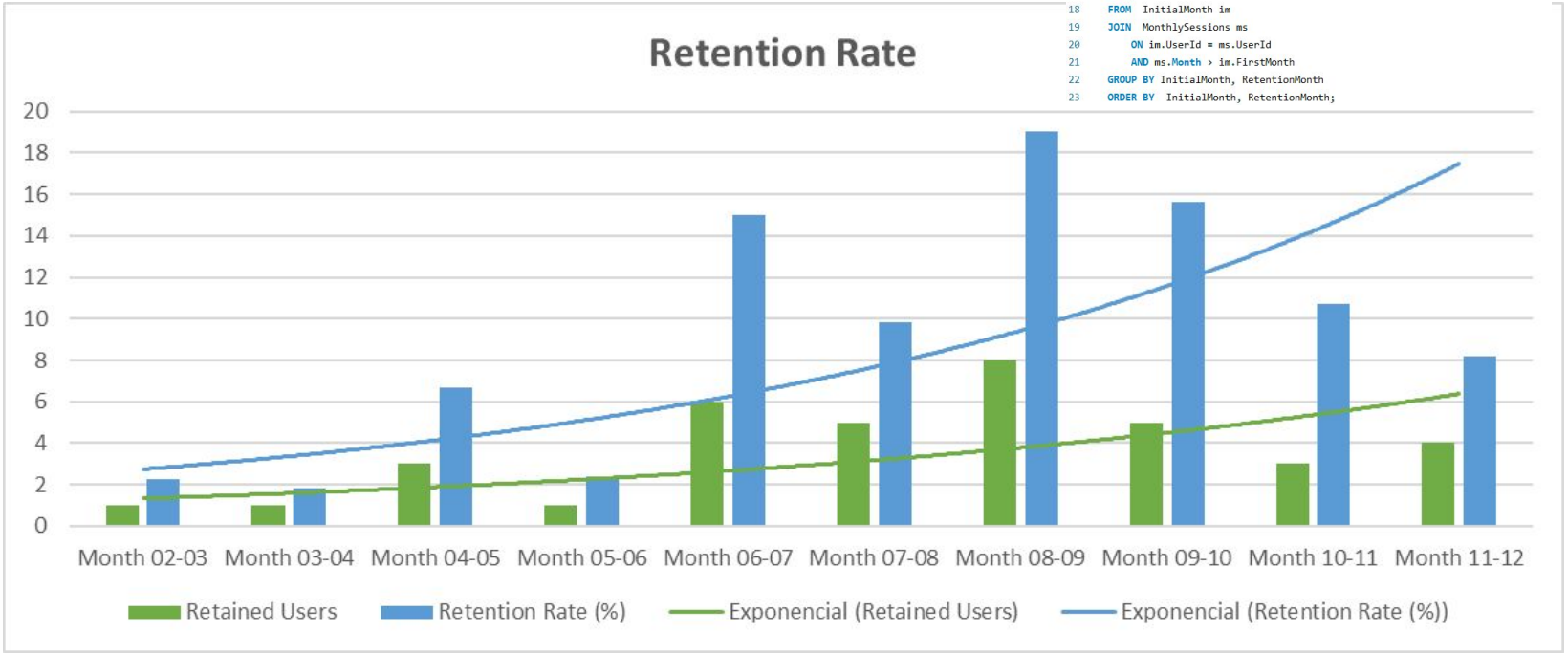
Stickiness



Visualize & Report

Retention Rate

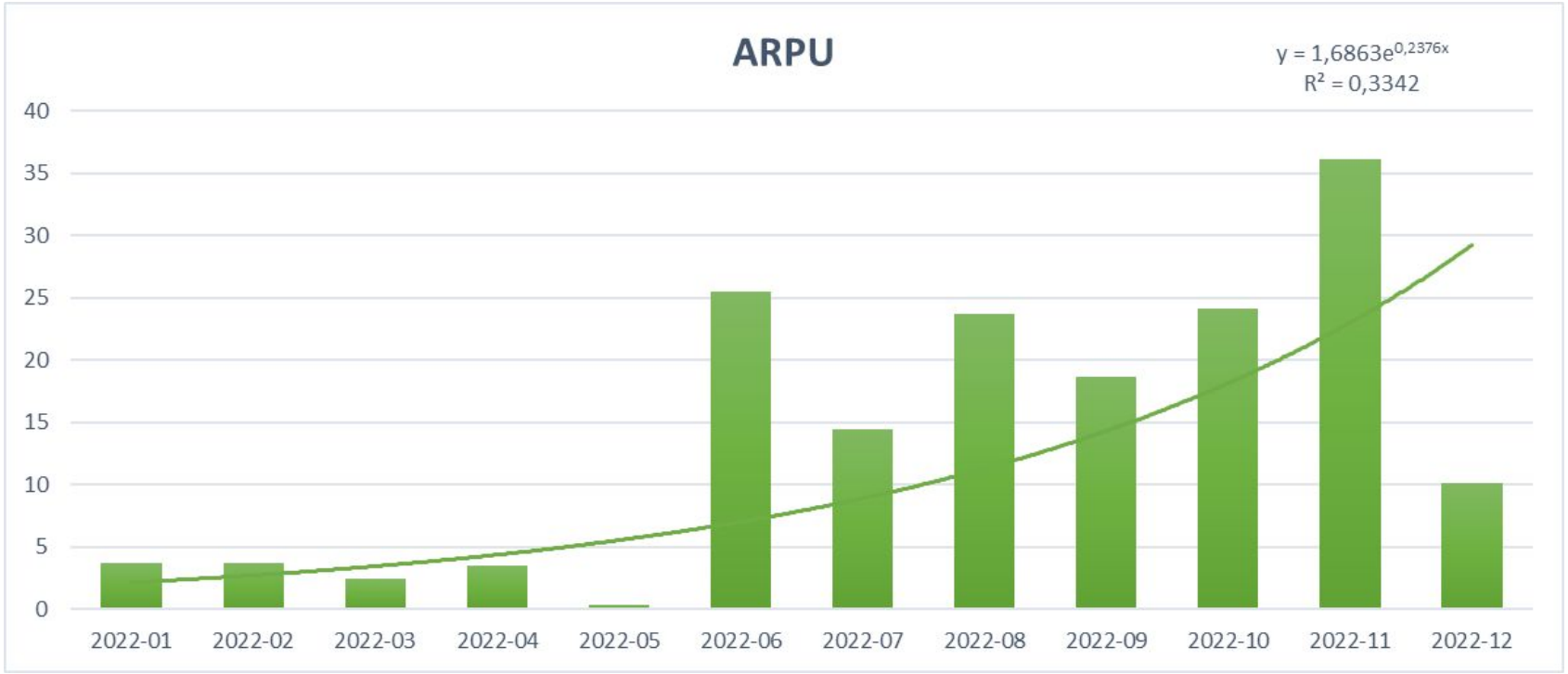
```sql
WITH InitialMonth AS (
    SELECT UserId, MIN(DATE_FORMAT(StartSession, '%Y-%m')) AS FirstMonth
    FROM SessionsData
    GROUP BY UserId
),
MonthlySessions AS (
    SELECT UserId, DATE_FORMAT(StartSession, '%Y-%m') AS Month
    FROM SessionsData
    GROUP BY UserId, Month
)
SELECT
    im.FirstMonth AS InitialMonth,
    ms.Month AS RetentionMonth,
    COUNT(DISTINCT ms.UserId) AS RetainedUsers,
    (COUNT(DISTINCT ms.UserId) /
        (SELECT COUNT(*) FROM InitialMonth WHERE FirstMonth = im.FirstMonth)
    ) * 100 AS RetentionRate
FROM  InitialMonth im
JOIN  MonthlySessions ms
    ON im.UserId = ms.UserId
    AND ms.Month > im.FirstMonth
GROUP BY InitialMonth, RetentionMonth
ORDER BY  InitialMonth, RetentionMonth;
```



Visualize & Report

ARPU

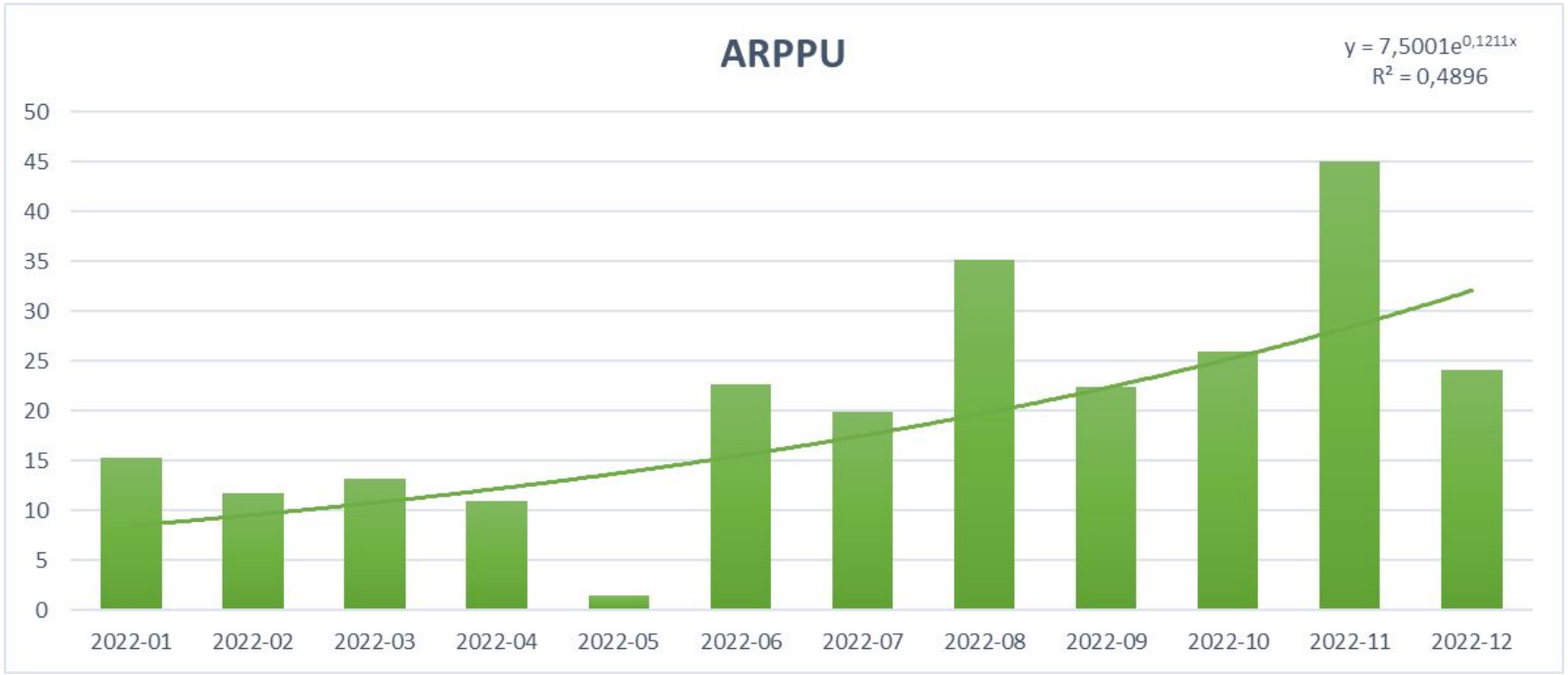
```sql
SELECT
    DATE_FORMAT(U.Install_Date, '%Y-%m') AS Month,
    (SUM(I.Price) / COUNT(DISTINCT U.ID)) AS ARPU
FROM
    UsersInfo U
LEFT JOIN
    SessionsData S ON U.ID = S.UserId
LEFT JOIN
    Purchases P ON S.id = P.sessionId
LEFT JOIN
    Items I ON P.itemId = I.Id
GROUP BY
    Month
ORDER BY
    Month;
```



Visualize & Report

ARPPU

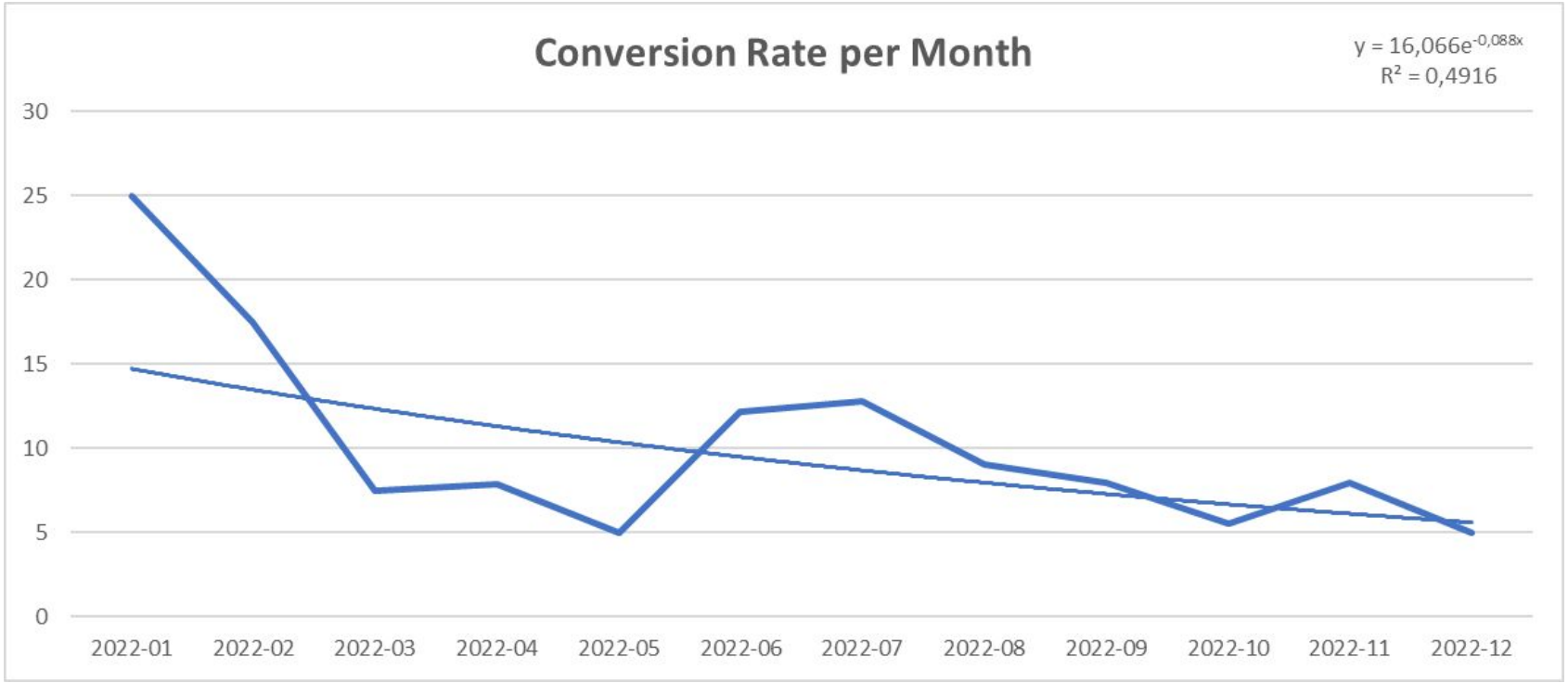
```sql
SELECT
    DATE_FORMAT(S.StartSession, '%Y-%m') AS Month,
    (SUM(I.Price) / COUNT(DISTINCT S.UserId)) AS ARPPU
FROM
    SessionsData S
INNER JOIN
    Purchases P ON S.id = P.sessionId
INNER JOIN
    Items I ON P.itemId = I.Id
GROUP BY
    Month
ORDER BY
    Month;
```



Visualize & Report

Conversion Rate

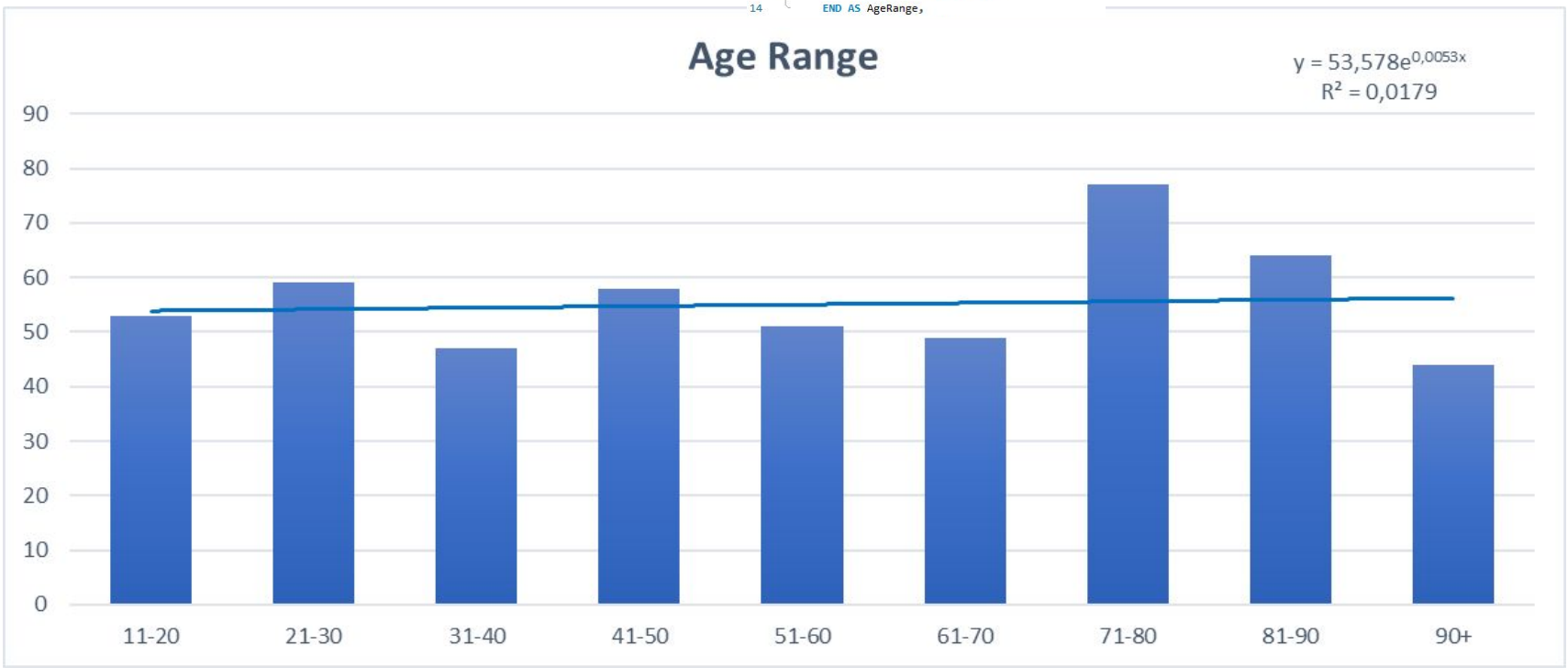
```sql
SELECT
    DATE_FORMAT(p.buyDate, '%Y-%m') AS Month,
    COUNT(DISTINCT sd.UserId) /
    (SELECT COUNT(*)
     FROM UsersInfo
     WHERE Install_Date <= LAST_DAY(p.buyDate)) * 100 AS Conversion_Rate
FROM  Purchases p
JOIN SessionsData sd ON p.sessionId = sd.id
GROUP BY Month
ORDER BY Month;
```



Visualize & Report

Users Age

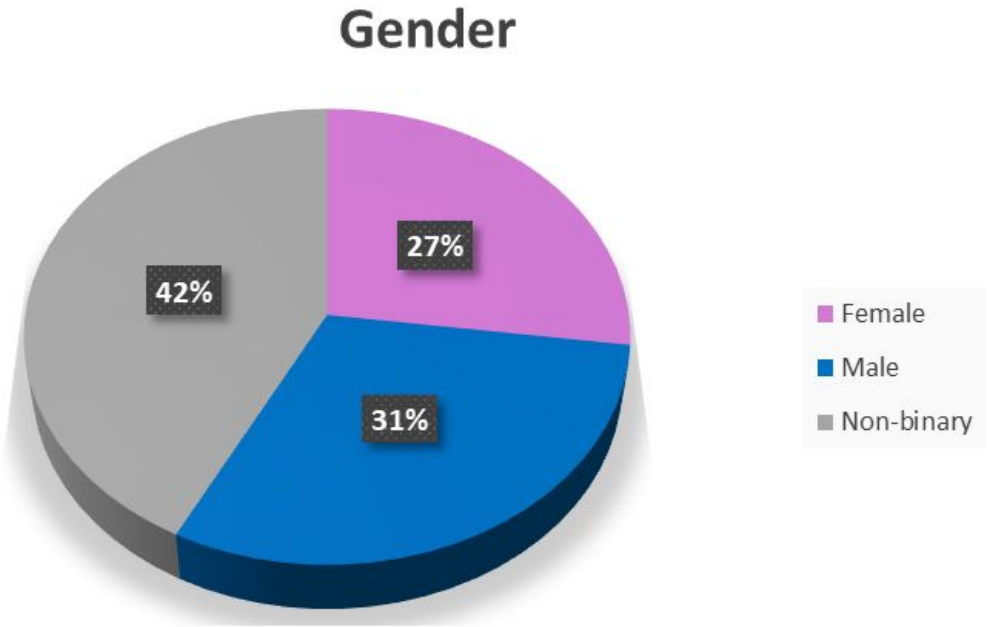
```sql
SELECT
    CASE
        WHEN Age IS NULL THEN 'No age provided'
        WHEN Age BETWEEN 0 AND 10 THEN '0-10'
        WHEN Age BETWEEN 11 AND 20 THEN '11-20'
        WHEN Age BETWEEN 21 AND 30 THEN '21-30'
        WHEN Age BETWEEN 31 AND 40 THEN '31-40'
        WHEN Age BETWEEN 41 AND 50 THEN '41-50'
        WHEN Age BETWEEN 51 AND 60 THEN '51-60'
        WHEN Age BETWEEN 61 AND 70 THEN '61-70'
        WHEN Age BETWEEN 71 AND 80 THEN '71-80'
        WHEN Age BETWEEN 81 AND 90 THEN '81-90'
        WHEN Age > 90 THEN '90+'
    END AS AgeRange,
    COUNT(Id) AS UserCount
FROM
    UsersInfo
GROUP BY AgeRange
ORDER BY AgeRange;
```



Visualize & Report

Users Gender

```sql
SELECT
    CASE
        WHEN Gender IS NULL THEN 'No gender provided'
        WHEN Gender <= 0.3 THEN 'Female'
        WHEN Gender >= 0.7 THEN 'Male'
        ELSE 'Non-binary'
    END AS GenderCategory,
    COUNT(Id) AS UserCount
FROM
    UsersInfo
GROUP BY GenderCategory
ORDER BY GenderCategory;
```



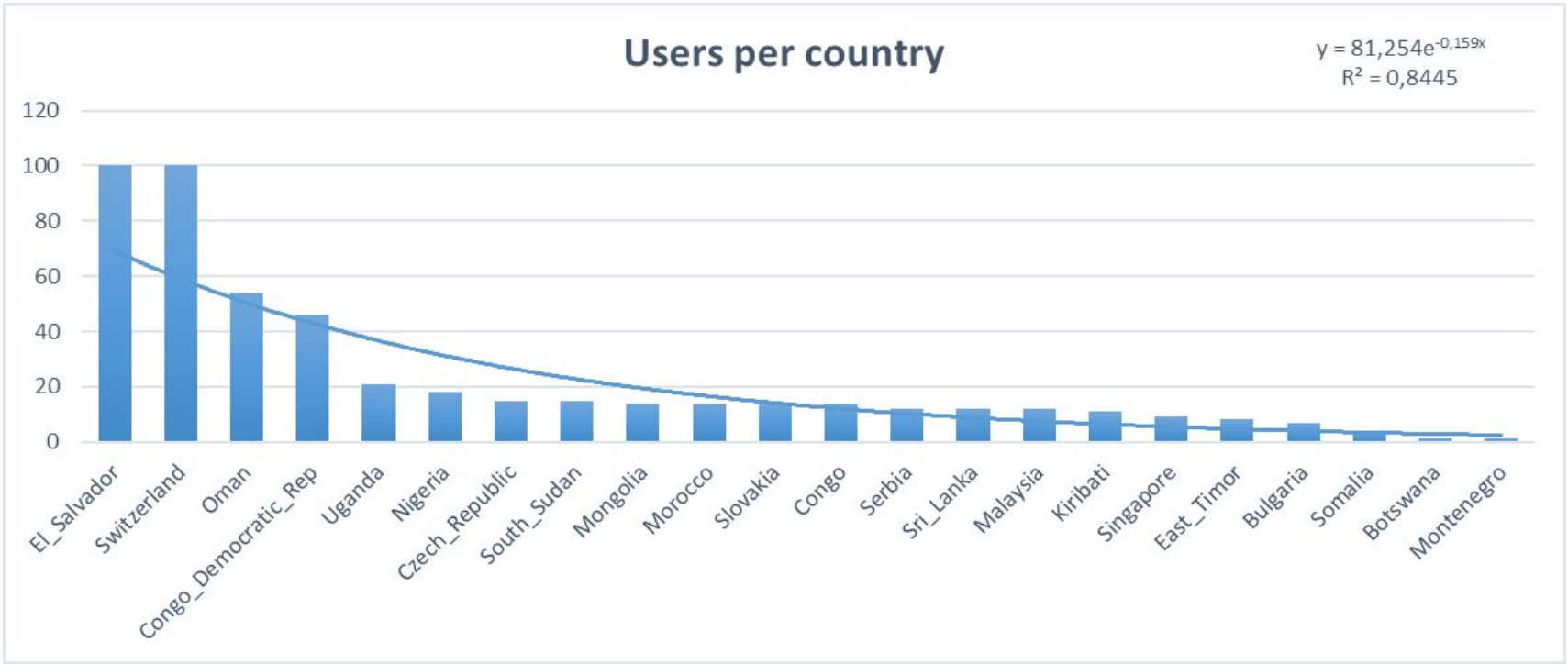
Female	136
Male	153
Non-binary	213

0.3 <= Female
Between 0.3 & 0.7 = Non-binary
0.7 >= Male

Visualize & Report

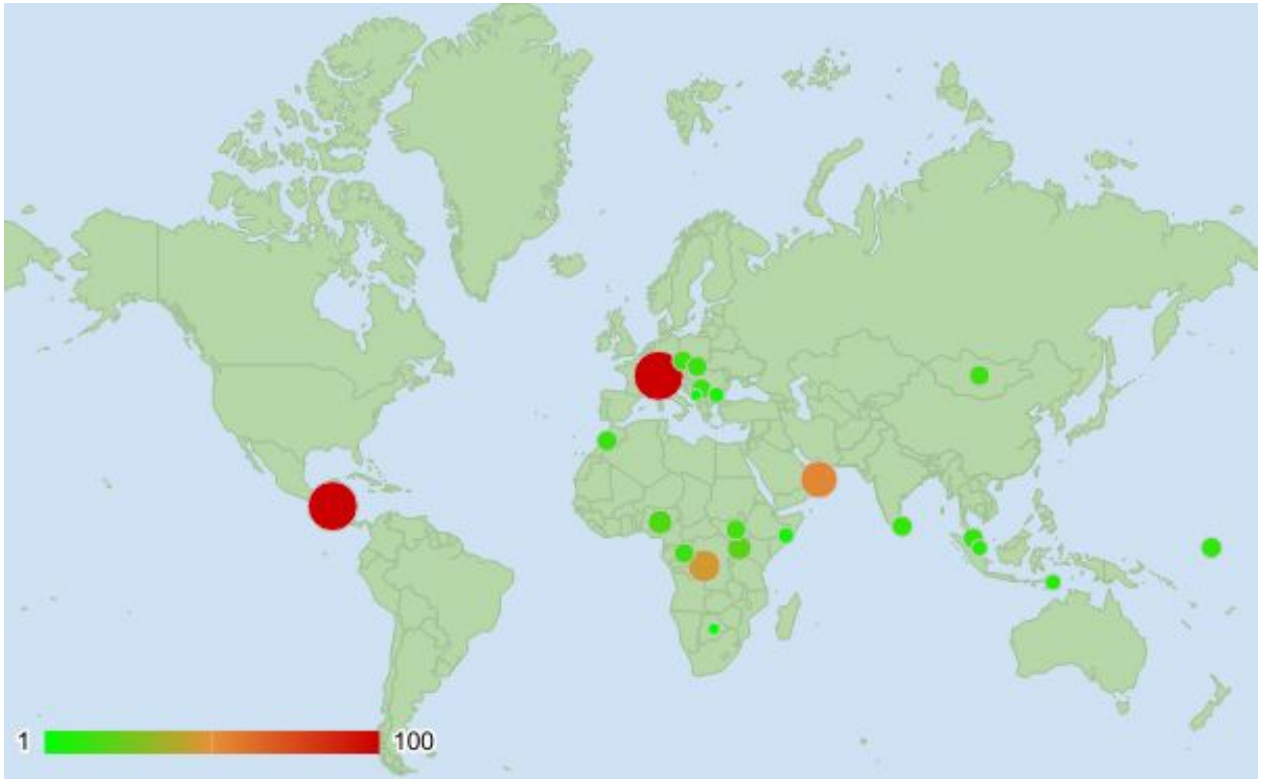
Users Country

```sql
SELECT
    Country,
    COUNT(DISTINCT Id) AS UserCount
FROM
    UsersInfo
GROUP BY Country
ORDER BY UserCount DESC;
```



Visualize & Report

Users Country



Visualize & Report

Extra data

Most profitable Item & Best-Selling Item

ItemId	Price	PurchaseCount	TotalRevenue
5	99.99	47	4699.529899597168
3	9.99	111	1108.8899745941162
2	1.99	180	358.20000171661377
1	0.99	352	348.4800033569336
4	49.99	6	299.9400100708008

ItemId	Price	PurchaseCount	TotalRevenue
1	0.99	352	348.4800033569336
2	1.99	180	358.20000171661377
3	9.99	111	1108.8899745941162
5	99.99	47	4699.529899597168
4	49.99	6	299.9400100708008

Avg time session: 527,54s

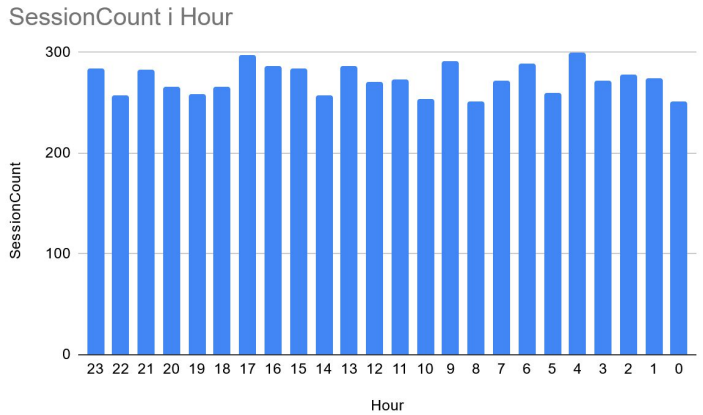
Top 5 Active Users

UserId	SessionCount
436	106
271	103
452	92
144	91
140	88

Top 10 Most Played Days

PlayDate	SessionCount
2022-11-12	56
2022-09-05	55
2022-12-12	54
2022-09-10	50
2022-07-16	49
2022-11-13	49
2022-09-04	48
2022-09-06	47
2022-12-11	46
2022-08-04	46

Most Played Hours



Gracias