Delivery 1

By Albert Chica & Marta Llurba

Index

1.

Code Structure

- Data_Transmission
- Php Files

2.

KPI

- DAU/MAU
- User Activity
- Purchases
- User Engagement
- Session Analysis
- ARPU & ARPPU

Code Structure

Data_Transmission.cs

```
using System;
using System.Collections;
using UnityEngine;
using UnityEngine.Networking;

public class DataTransmission : MonoBehaviour
{
    uint currentUserId;
    uint currentSessionId;
    uint currentPurchaseId;

    private enum ActionType { NewPlayer, StartSession, EndSession, BuyItem }

    private void OnEnable()
    {
        Simulator.OnNewPlayer += (name, country, date) =>
        {
            WWWForm form = CreateForm(ActionType.NewPlayer, name, country, date.ToString("yyyy-MM-dd HH:mm:ss"));
            StartCoroutine(UploadData(ActionType.NewPlayer, form, "https://citmalumnes.upc.es/~albertcf5/Player_Data.php"));
        };

        Simulator.OnNewSession += (date) =>
        {
            WWWForm form = CreateForm(ActionType.StartSession, date: date.ToString("yyyy-MM-dd HH:mm:ss"));
            StartCoroutine(UploadData(ActionType.StartSession, form, "https://citmalumnes.upc.es/~albertcf5/Session_Data.php"));
        };

        Simulator.OnEndSession += (date) =>
        {
            WWWForm form = CreateForm(ActionType.EndSession, date: date.ToString("yyyy-MM-dd HH:mm:ss"));
            StartCoroutine(UploadData(ActionType.EndSession, form, "https://citmalumnes.upc.es/~albertcf5/Close_Session_Data.php"));
        };

        Simulator.OnBuyItem += (item, date) =>
        {
            WWWForm form = CreateForm(ActionType.BuyItem, date: date.ToString("yyyy-MM-dd HH:mm:ss"), item: item);
            StartCoroutine(UploadData(ActionType.BuyItem, form, "https://citmalumnes.upc.es/~albertcf5/Purchase_Data.php"));
        };
    }
```

Data_Transmission.cs

```
private WWWForm CreateForm(ActionType actionType, string name = null, string country = null, string date = null, int item = 0)
{
    WWWForm form = new WWWForm();
    switch (actionType)
    {
        case ActionType.NewPlayer:
            form.AddField("Name", name);
            form.AddField("Country", country);
            form.AddField("Date", date);
            break;

        case ActionType.StartSession:
            form.AddField("User_ID", currentUserId.ToString());
            form.AddField("Start_Session", date);
            break;

        case ActionType.EndSession:
            form.AddField("User_ID", currentUserId.ToString());
            form.AddField("End_Session", date);
            form.AddField("Session_ID", currentSessionId.ToString());
            break;

        case ActionType.BuyItem:
            form.AddField("Item", item.ToString());
            form.AddField("User_ID", currentUserId.ToString());
            form.AddField("Session_ID", currentSessionId.ToString());
            form.AddField("Buy_Date", date);
            break;
    }
    return form;
}
```

Data_Transmission.cs

```
IEnumerator UploadData(ActionType actionType, WWWForm form, string url)
{
    using (UnityWebRequest www = UnityWebRequest.Post(url, form))
    {
        yield return www.SendWebRequest();

        if (www.result != UnityWebRequest.Result.Success)
        {
            UnityEngine.Debug.LogError($"{actionType} data upload failed: " + www.error);
        }
        else
        {
            string answer = www.downloadHandler.text.Trim(new char[] { '\uFEFF', '\u200B', ' ', '\t', '\r', '\n' });

            if (actionType == ActionType.NewPlayer && uint.TryParse(answer, out uint parsedUserId) && parsedUserId > 0)
            {
                currentUserId = parsedUserId;
                CallbackEvents.OnAddPlayerCallback.Invoke(currentUserId);
            }
            else if (actionType == ActionType.StartSession && uint.TryParse(answer, out uint parsedSessionId) && parsedSessionId > 0)
            {
                currentSessionId = parsedSessionId;
                CallbackEvents.OnNewSessionCallback.Invoke(currentSessionId);
            }
            else if (actionType == ActionType.EndSession)
            {
                CallbackEvents.OnEndSessionCallback.Invoke(currentSessionId);
            }
            else if (actionType == ActionType.BuyItem && uint.TryParse(answer, out uint parsedPurchaseId) && parsedPurchaseId > 0)
            {
                currentPurchaseId = parsedPurchaseId;
                CallbackEvents.OnItemBuyCallback.Invoke();
            }
            else
            {
                UnityEngine.Debug.LogError($"Invalid response for {actionType}: " + answer);
            }
        }
    }
}
```

```php
<?php
$servername = "localhost:3306";
$username = "albertcf5";
$password = "48103884m";
$database = "albertcf5";

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

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

db_connection.php

```php
<?php
include 'db_connect.php';

$name = $_POST["Name"];
$country = $_POST["Country"];
$date = $_POST["Date"];

error_log("Received player data: Name={$name}, Country={$country}, Date={$date}");

$stmt = $conn->prepare("INSERT INTO `Players`(`Name`, `Country`, `Date`) VALUES (?, ?, ?)");
$stmt->bind_param("sss", $name, $country, $date);

if ($stmt->execute()) {
    echo $conn->insert_id;
} else {
    error_log("Error in Player_Data.php: " . $stmt->error);
    echo "Error: " . $stmt->error;
}

$stmt->close();
$conn->close();
?>
```

Player_Data.php

```php
<?php
include 'db_connect.php';

$userId = $_POST["User_ID"];
$sessionId = $_POST["Session_ID"];
$itemId = $_POST["Item"];
$buyDate = $_POST["Buy_Date"];

error_log("Received purchase data: User_ID={$userId}, Session_ID={$sessionId}, Item={$itemId}, Buy_Date={$buyDate}");

$stmt = $conn->prepare("INSERT INTO `Purchases`(`userId`, `sessionId`, `itemId`, `buyDate`) VALUES (?, ?, ?, ?)");
$stmt->bind_param("iiis", $userId, $sessionId, $itemId, $buyDate);

if ($stmt->execute()) {
    echo $conn->insert_id;
} else {
    error_log("Error in Purchase_Data.php: " . $stmt->error);
    echo "Error: " . $stmt->error;
}

$stmt->close();
$conn->close();
?>
```

Purchases_Data.php

```php
<?php
include 'db_connect.php';

$userId = $_POST["User_ID"];
$startSession = $_POST["Start_Session"];

error_log("Received session start data: User_ID={$userId}, Start_Session={$startSession}");

$stmt = $conn->prepare("INSERT INTO `Sessions`(`userId`, `startSession`) VALUES (?, ?)");
$stmt->bind_param("is", $userId, $startSession);

if ($stmt->execute()) {
    echo $conn->insert_id;
} else {
    error_log("Error in Session_Data.php: " . $stmt->error);
    echo "Error: " . $stmt->error;
}

$stmt->close();
$conn->close();
?>
```

Session_Data.php

```php
<?php
include 'db_connect.php';

$sessionId = $_POST["Session_ID"];
$endSession = $_POST["End_Session"];

error_log("Received end session data: Session_ID={$sessionId}, End_Session={$endSession}");

$stmt = $conn->prepare("UPDATE `Sessions` SET `endSession` = ? WHERE `sessionId` = ?");
$stmt->bind_param("si", $endSession, $sessionId);

if ($stmt->execute()) {
    if ($stmt->affected_rows > 0) {
        //echo "Session closed successfully";
        echo $endSession;
    } else {
        error_log("No session updated in Close_Session_Data.php");
        echo "No session updated";
    }
} else {
    error_log("Error in Close_Session_Data.php: " . $stmt->error);
    echo "Error: " . $stmt->error;
}

$stmt->close();
$conn->close();
?>
```

Close_Session_Data.php

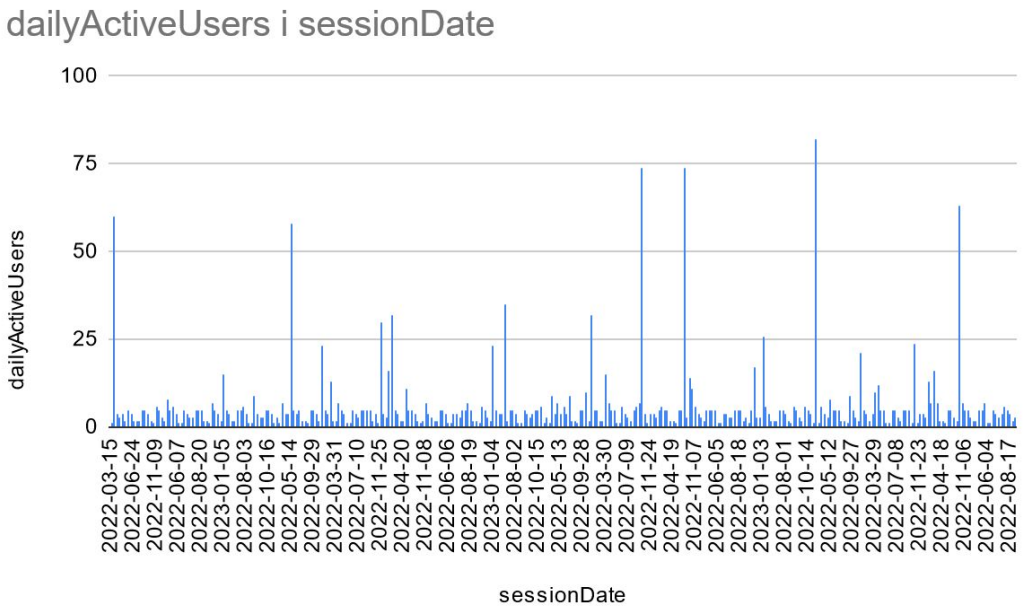
KPIs

```sql
SELECT DATE(s.startSession) AS sessionDate,
COUNT(DISTINCT s.userId) AS dailyActiveUsers,
SUM(COUNT(DISTINCT s.userId)) OVER() AS MonthlyActiveUsers,
COUNT(DISTINCT s.userId) * 100 / SUM(COUNT(DISTINCT s.userId)) OVER() AS stickiness
FROM Sessions s WHERE s.startSession BETWEEN '2022-09-01' AND '2022-09-31'
GROUP BY DATE(s.startSession)
ORDER BY dailyActiveUsers DESC
```

KPIs

sessionDate	dailyActiveUsers ▼ 1	MonthlyActiveUsers	stickiness
2022-09-24	17	144	11.8056
2022-09-25	5	144	3.4722
2022-09-29	5	144	3.4722
2022-09-26	5	144	3.4722
2022-09-06	5	144	3.4722
2022-09-03	5	144	3.4722
2022-09-30	5	144	3.4722
2022-09-10	5	144	3.4722
2022-09-27	5	144	3.4722
2022-09-07	5	144	3.4722
2022-09-01	5	144	3.4722
2022-09-28	5	144	3.4722
2022-09-18	4	144	2.7778
2022-09-08	4	144	2.7778
2022-09-15	4	144	2.7778
2022-09-05	4	144	2.7778
2022-09-22	4	144	2.7778
2022-09-12	4	144	2.7778
2022-09-02	4	144	2.7778
2022-09-19	4	144	2.7778

DAU

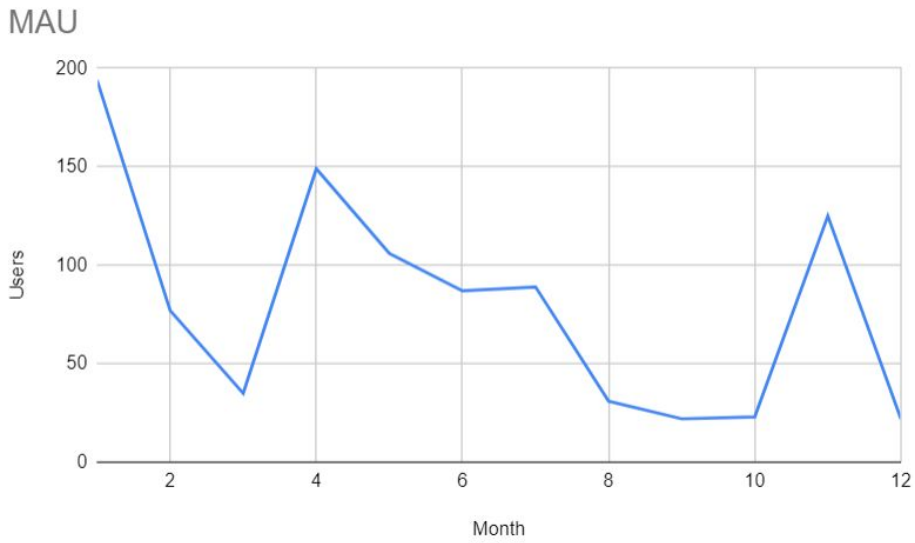


KPIs

MAU

```sql
SELECT YEAR(startSession) as year,
MONTH(startSession) as month,
COUNT(DISTINCT userId) as monthlyUsers
FROM Sessions
GROUP BY YEAR(startSession), MONTH(startSession)
ORDER BY YEAR(startSession), MONTH(startSession);
```

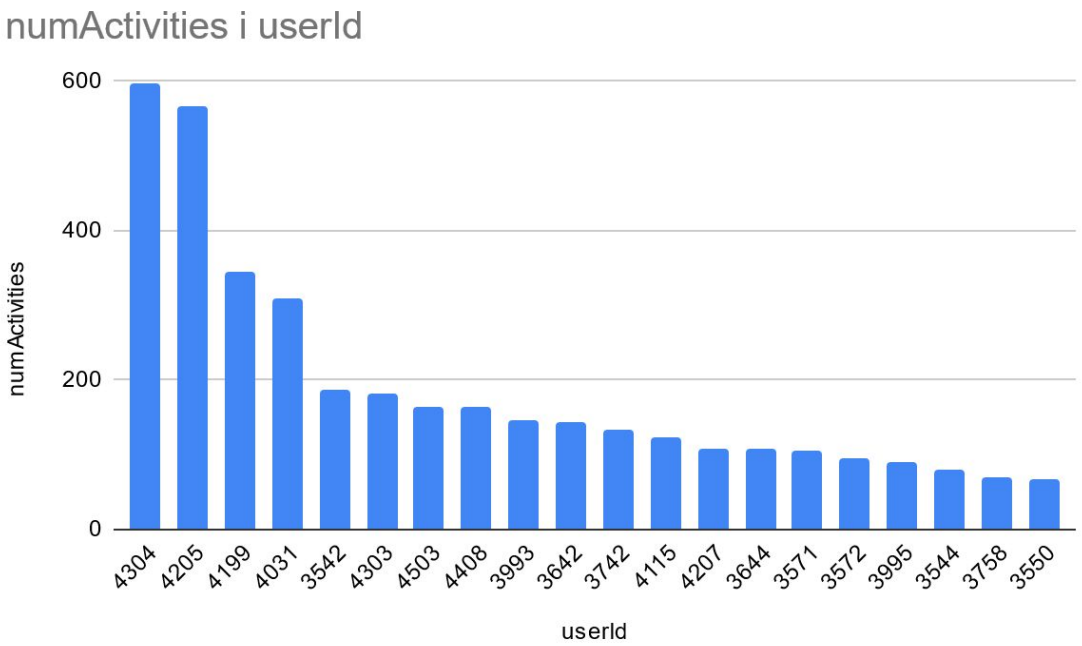
year	month	monthlyUsers
2022	2	77
2022	3	35
2022	4	149
2022	5	106
2022	6	87
2022	7	89
2022	8	31
2022	9	22
2022	10	23
2022	11	115
2022	12	22
2023	1	194



```sql
SELECT userId, COUNT(sessionId) AS numActivities
FROM Sessions
GROUP BY userId
ORDER BY numActivities DESC
```

KPIs

User Activity - Number of Session per User



userId	numActivities ▼ 1
4304	597
4205	566
4199	345
4031	310
3542	186
4303	183
4503	165
4408	163
3993	147
3642	143
3742	133
4115	124
4207	108
3644	108
3571	105
3572	96
3995	90
3544	81
3758	71
3550	68

KPIs

User Activity – Average Session Duration

	user	timeSession
Edit Copy Delete	4377	51129
Edit Copy Delete	4303	17922
Edit Copy Delete	4303	17715
Edit Copy Delete	4303	17644
Edit Copy Delete	4303	17174
Edit Copy Delete	4114	16640
Edit Copy Delete	4115	16530
Edit Copy Delete	4115	16429
Edit Copy Delete	4303	16264
Edit Copy Delete	4115	16248
Edit Copy Delete	4303	16054
Edit Copy Delete	4115	16001
Edit Copy Delete	4115	15738
Edit Copy Delete	4303	15704
Edit Copy Delete	4115	15696
Edit Copy Delete	4115	15627
Edit Copy Delete	4115	15473

```sql
SELECT DISTINCT userId as user,
TIMESTAMPDIFF(MINUTE, startSession, endSession) as timeSession
FROM Sessions
WHERE startSession IS NOT NULL
AND endSession IS NOT NULL
ORDER BY timeSession DESC
```

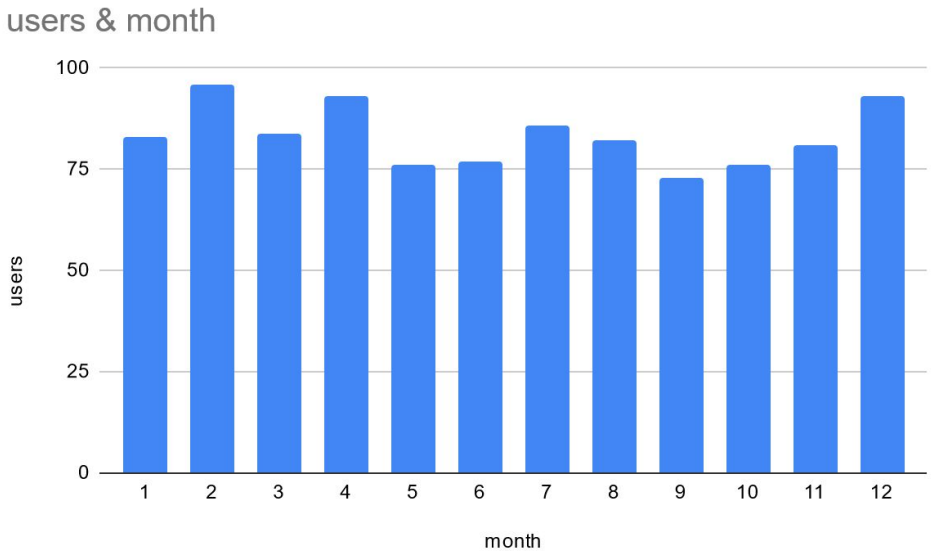
```sql
SELECT AVG(TIMESTAMPDIFF(MINUTE, startSession, endSession)) AS averageTime
FROM Sessions
WHERE startSession IS NOT NULL
AND endSession IS NOT NULL
```

averageTime
2969.3862

KPIs

User Activity - New Player Acquisition Rate

```sql
SELECT YEAR(Date) AS year,
MONTH(Date) AS month,
COUNT(userId) AS users
FROM Players
GROUP BY month
ORDER BY users DESC
```



KPIs

Purchases - Average Purchase Value

```sql
SELECT COUNT(DISTINCT p.purchaseId) as purchases,
SUM(Price) as totalIncome,
SUM(Price)/COUNT(DISTINCT p.purchaseId) as averagePurchaseValue
FROM Purchases p
join Items i on p.itemId = i.Id
```

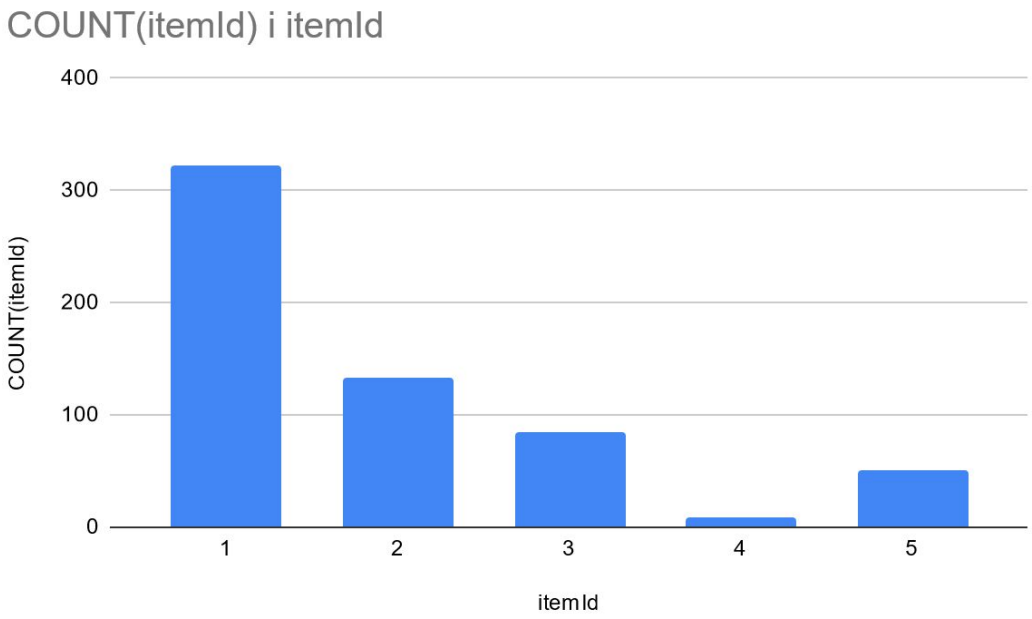
purchases	totalIncome	averagePurchaseValue
598	6922.019889593124	11.575284096309572

```sql
SELECT itemId,
COUNT(itemId)
FROM Purchases
GROUP BY itemId
```

KPIs

Purchases – Popular Items

itemId	COUNT(itemId)
1	322
2	133
3	84
4	8
5	51



KPIs

User Engagement – Conversion Rate from Sessions to Purchases

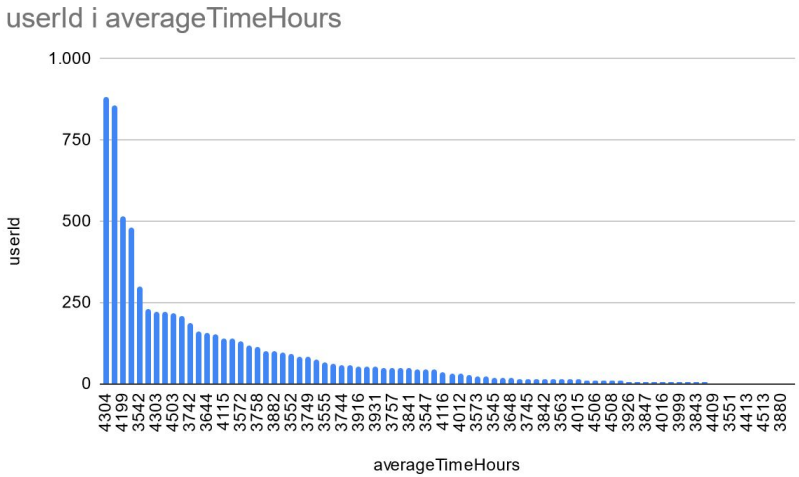
```sql
SELECT COUNT(DISTINCT s.sessionId) as totalSessions,
COUNT(DISTINCT p.sessionId) as purchaseSessions,
COUNT(DISTINCT p.sessionId) * 100/COUNT(DISTINCT s.sessionId) as convertionRate
FROM Sessions s
LEFT JOIN Purchases p ON s.sessionId = p.sessionId
```

totalSessions	purchaseSessions	convertionRate
6056	598	9.8745

KPIs

Session Analysis - Average Time Between Sessions

```sql
SELECT s1.userId,
AVG(TIMESTAMPDIFF(HOUR, s1.endSession, s2.startSession)) AS averageTimeHours
FROM Sessions s1
JOIN Sessions s2 ON s1.userId = s2.userId
AND s1.startSession < s2.startSession
WHERE s1. endSession IS NOT NULL AND
s2.startSession IS NOT NULL AND
s1.endSession < s2.startSession
GROUP BY s1.userId
ORDER BY averageTimeHours DESC
```



userId	averageTimeHours ▼ 1
4304	820.5707
4205	797.3257
4031	462.9093
4199	419.5220
3542	293.4203
3993	222.1917
3642	202.2296
3742	181.5119
4408	177.5778
4503	165.0427

KPIs

ARPU & ARPPU

playersThatBought	totalPurchases	totalPlayers	ARPU	ARPPU
109	598	1000	6.922019889593124	63.50476962929472

```sql
SELECT
COUNT(DISTINCT p.userId) AS playersThatBought,
COUNT(p.userId) as totalPurchases,
COUNT(DISTINCT pl.userId) as totalPlayers,
SUM(Price) / COUNT(DISTINCT pl.userId) as ARPU,
SUM(Price) / COUNT(DISTINCT p.userId) as ARPPU
FROM Purchases p
JOIN Items i ON p.itemId = i.Id
RIGHT JOIN Players pl on pl.userId = p.userId
ORDER BY p.purchaseId DESC
```