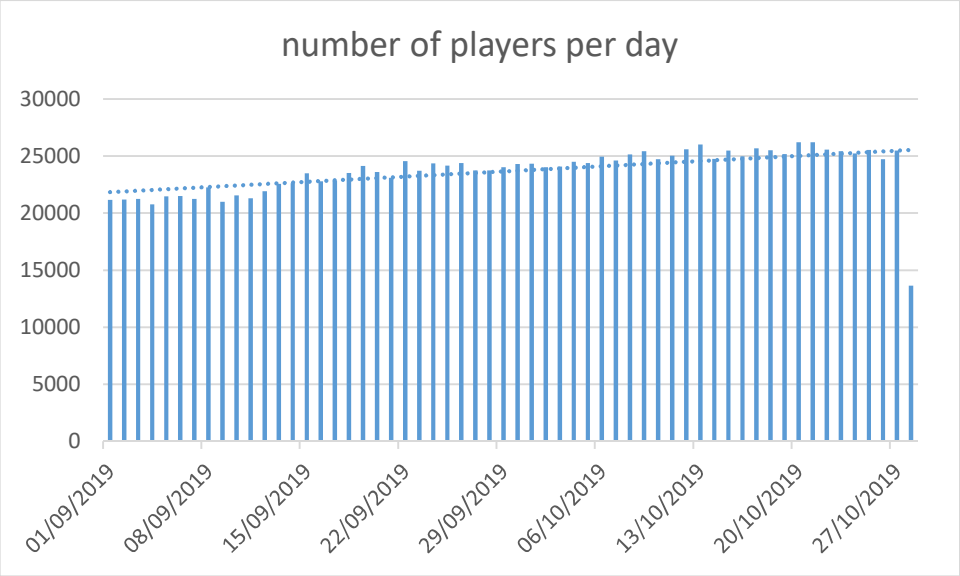
Analytical Section

1. User Activity

1.1 How many users are using the app every day?

- As mentioned in the Technical Section, Part I, I am sticking to the "De Dicto" interpretation of this sentence ("For each Day, what is the number of users?") and I am considering only the "open_app" events.

```
SELECT CAST(event_ts AS DATE) AS "Day",
COUNT(DISTINCT user_id) AS "Number of Players"
FROM retention
WHERE Day BETWEEN '2019-09-01' AND '2019-10-31' -- OPTIONAL, in every
request since the database stops on the 10-28
/*OPTIONAL*/AND event_name = 'open_app'
GROUP BY Day
ORDER BY Day;
```



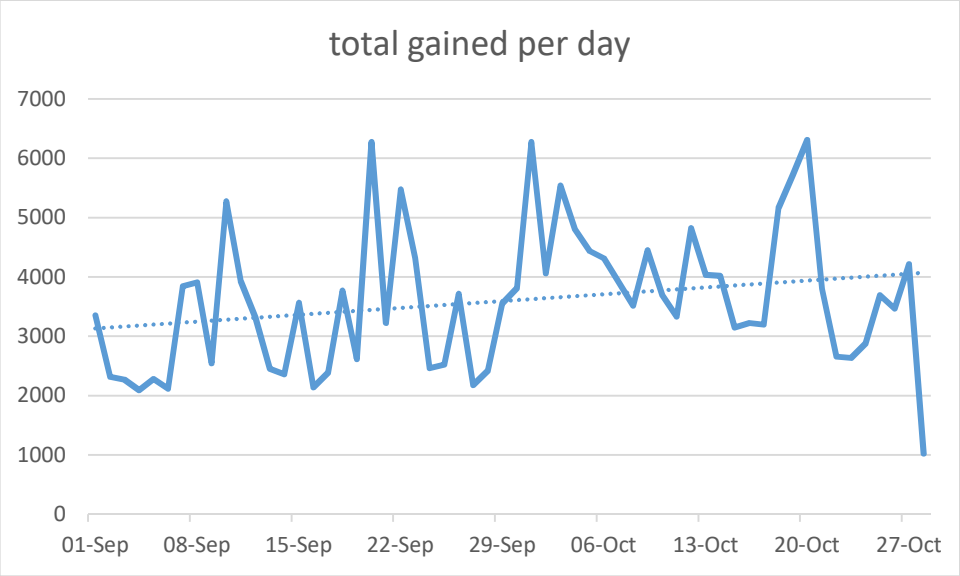
The trend seems to suggest a growth with very little variation day by day. This made me think that the sharp decrease of number of players for the 28th of October had to be explained out by external reasons. Checking the database for incongruences I confirmed my first theory: the table only has data up until 2 o'clock PM for that day. The reason why this occurs may be investigated further.
No particularly relevant pattern based on weekends or working-day seems to emerge, only small alterations.

1.2 How much do the users spend buying IAP every day?

```
SELECT to_char(CAST(event_ts AS DATE), 'dd-mm') AS "Date",
SUM(sku_price) AS "Total Gained per Day"
FROM purchases
```

```
GROUP BY date
ORDER BY date;
```

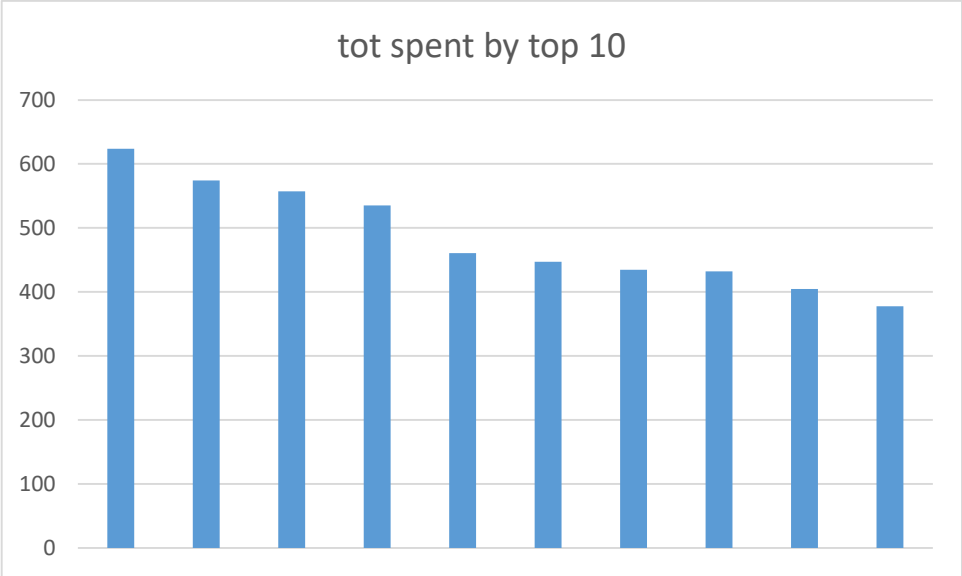
The total spent in iap purchases is much more variable day by day than the number of players.
However, it is noticeable that it follows the same trend.
Checking on the calendar of September and October 2019 the peaks and valleys of the function do not seem to be particularly related with workdays or weekends.



1.3 How much did the top 10 spenders spend through both September and October?

```
SELECT user_id,
SUM(sku_price) as tot_spent
FROM purchases
GROUP BY user_id
ORDER BY tot_spent DESC
LIMIT 10;
```

The average amount spent by the top 10 spenders is 480€ circa. This number seemed particularly high, so I confronted the Top 10 with the entire list.
The average spent by the Top 10 spenders is much greater than the average spent by every player (14,5€).
I will expand this point in the last question.



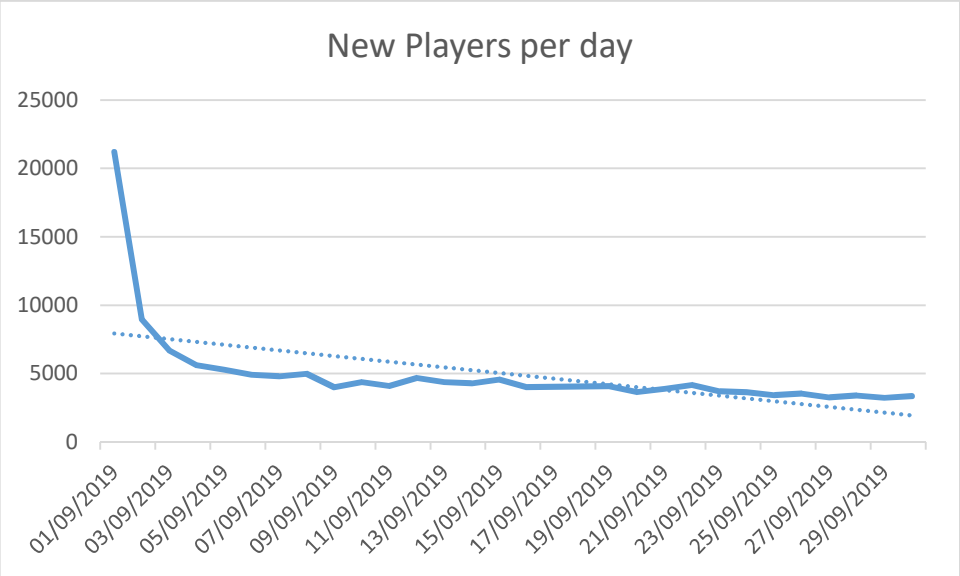
2. Retention

2.1 How many users joined on each day of September?

```
3. SELECT first_join, COUNT(DISTINCT user_id)
   FROM (
        SELECT user_id, MIN(CAST(event_ts AS DATE)) AS first_join
        FROM retention
        GROUP BY user_id

   )
   GROUP BY first_join
   ORDER BY first_join;
```

The first days in particular do not seem in line with the others, but we can immediately infer that it is because on the first day every player is counted as new player.
If the app launched the 1st of September this would not be a problem of course.
In any case it could be useful to eliminate this problem by adding another id_event such as "registration" or "first_join", in order to collect these data more accurately.
However, even not considering the first few days, there seem to be a negative trend on new users.



2.2 How many users that joined on the 15th of September used the app after 7 days?

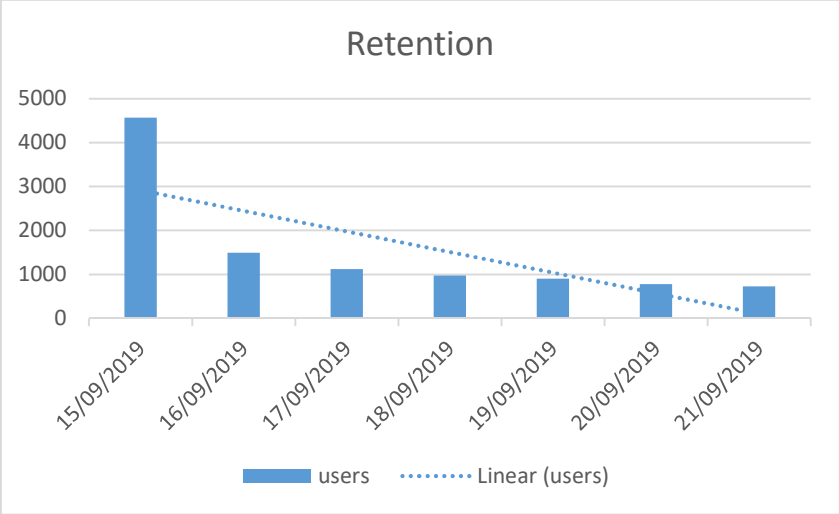
```
SELECT fj_date as Day,
Users as "Users",
Users || '/' || (
    SELECT MAX(Users)
    FROM (
          SELECT CAST(event_ts AS DATE) AS fj_date, COUNT(DISTINCT
players) AS Users
    FROM retention
    INNER JOIN  (
          SELECT user_id AS players, MIN(CAST(event_ts AS DATE)) AS
first_join
              FROM retention
              GROUP BY players
              HAVING MIN(CAST(event_ts AS DATE)) = '2019-09-15'
          ) AS sub
    ON retention.user_id = sub.players
    GROUP BY fj_date
    ORDER BY fj_date
          )
  ) AS "Calculation",
round((CAST(Users AS FLOAT )/(
    SELECT MAX(Users)
    FROM (
          SELECT CAST(event_ts AS DATE) AS fj_date, COUNT(DISTINCT
players) AS Users
    FROM retention
    INNER JOIN  (
          SELECT user_id AS players, MIN(CAST(event_ts AS DATE)) AS
first_join
              FROM retention
              GROUP BY players
```

```
            HAVING MIN(CAST(event_ts AS DATE)) = '2019-09-15'
          ) AS sub
    ON retention.user_id = sub.players
    GROUP BY fj_date
    ORDER BY fj_date
          )
  ))*100,
  2)||'%' AS "Retention"

FROM (
    SELECT CAST(event_ts AS DATE) AS fj_date, COUNT(DISTINCT  players) AS
Users
    FROM retention
    INNER JOIN  (
          SELECT user_id AS players, MIN(CAST(event_ts AS DATE)) AS
first_join
              FROM retention
              GROUP BY players
              HAVING MIN(CAST(event_ts AS DATE)) = '2019-09-15'
          ) AS sub
    ON retention.user_id = sub.players
    GROUP BY fj_date
    ORDER BY fj_date)
WHERE fj_date BETWEEN '2019-09-15' AND '2019-09-21'
GROUP BY fj_date, Users
ORDER BY fj_date
```

The retention in this particular interval follows in the first day a decrease much more steep than the one in the given example.

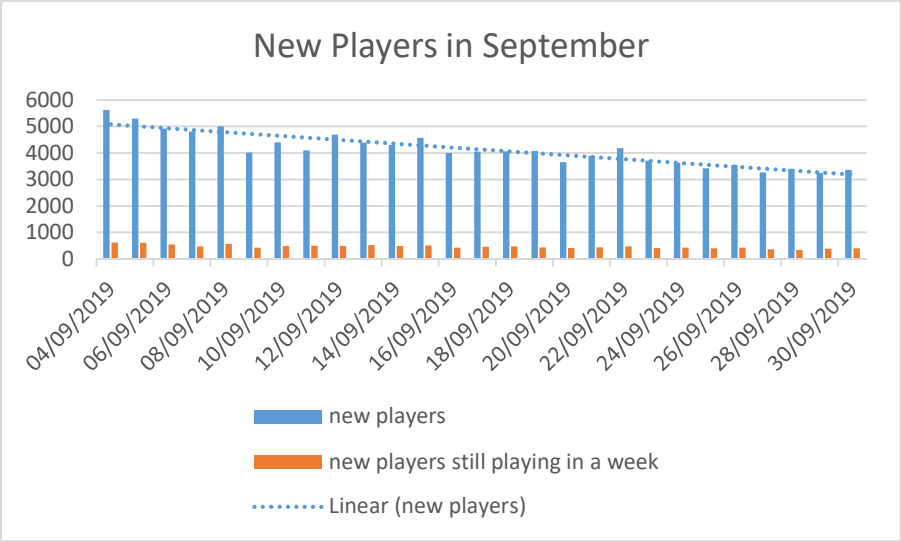
days	users	Retention
15/09/2019	4568	100%
16/09/2019	1490	32.62%
17/09/2019	1122	24.56%
18/09/2019	977	21.39%
19/09/2019	903	19.77%
20/09/2019	782	17.12%
21/09/2019	732	16.02%

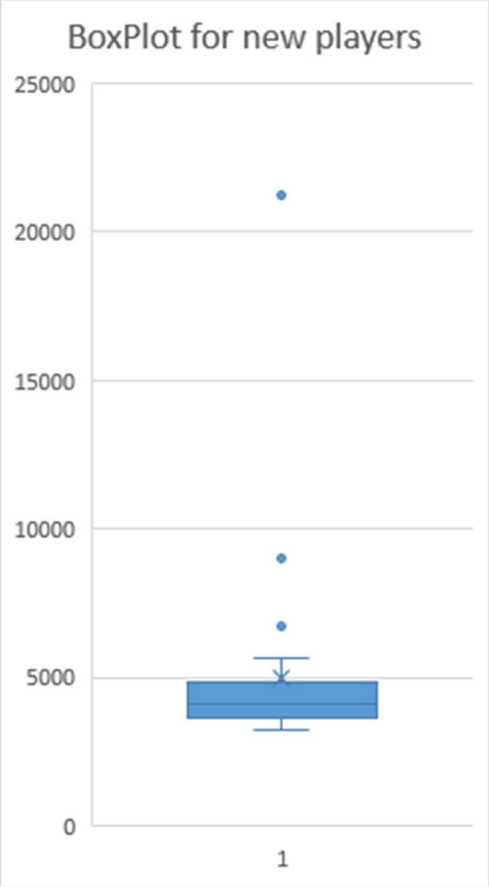


2.3 For each day of September how many users joined and how many of these users opened the app again after 7 days?

```
SELECT day, Player_Count "New Players", Player_Count_in_sev AS "New
Players still playing in a week", Round(CAST(Player_Count_in_sev AS FLOAT
)/Player_Count*100,2)||'%' AS "retention on day 7"
FROM
(
     SELECT first_join AS day, COUNT(DISTINCT user_id) AS Player_Count
     FROM (
          SELECT user_id, MIN(CAST(event_ts AS DATE)) AS first_join
          FROM retention
          WHERE CAST(event_ts AS DATE) BETWEEN '2019-09-01' AND '2019-
09-30'
          GROUP BY user_id

     )
     GROUP BY first_join
     ORDER BY first_join
     ) AS tab1
INNER JOIN (
     SELECT first_join, COUNT(DISTINCT player)  AS Player_Count_in_sev
     FROM
               (SELECT user_id AS new_player, MIN(CAST(event_ts AS
DATE)) AS first_join, first_join+13 AS sev_days_after
               FROM retention
               WHERE CAST(event_ts AS DATE) BETWEEN '2019-09-01' AND
'2019-09-30'
               GROUP BY user_id) as subtab1
          INNER JOIN (
          SELECT user_id AS player,CAST(event_ts AS DATE) as day
          FROM retention ) as subtab2
  ON subtab1.sev_days_after = subtab2.day
  WHERE subtab1.new_player = subtab2.player
  GROUP By first_join
  ORDER By first_join
  ) AS tab2
ON tab1.day = tab2.first_join
ORDER BY day
```

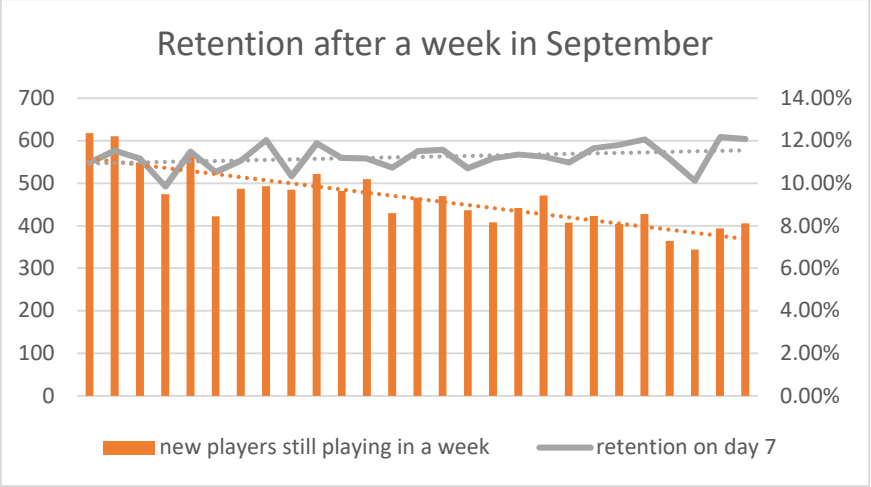




In this table the same problem of the point 2.1 arises: the first three days are particularly not representative. (ex. The new players on the first two days are outside the standard deviation, and the first three days are all outside the Box Plot).
In the table below these three days are not shown, so that the other data are more reliable and can stand out more. As we remember from 2.1, the trend of new users joining is negative. And, as better shown in the table below, the trend for new players still playing in a week is equally decreasing.

2.4 Using the values from the question above, what is the retention rate for each day of September. What's its trend?

The SQL code for this question is the same from the one before, as it was particularly handy to extract these data at the same time. When the first three days are removed once again, we can see that the trend for retention on day 7 is quite stable.

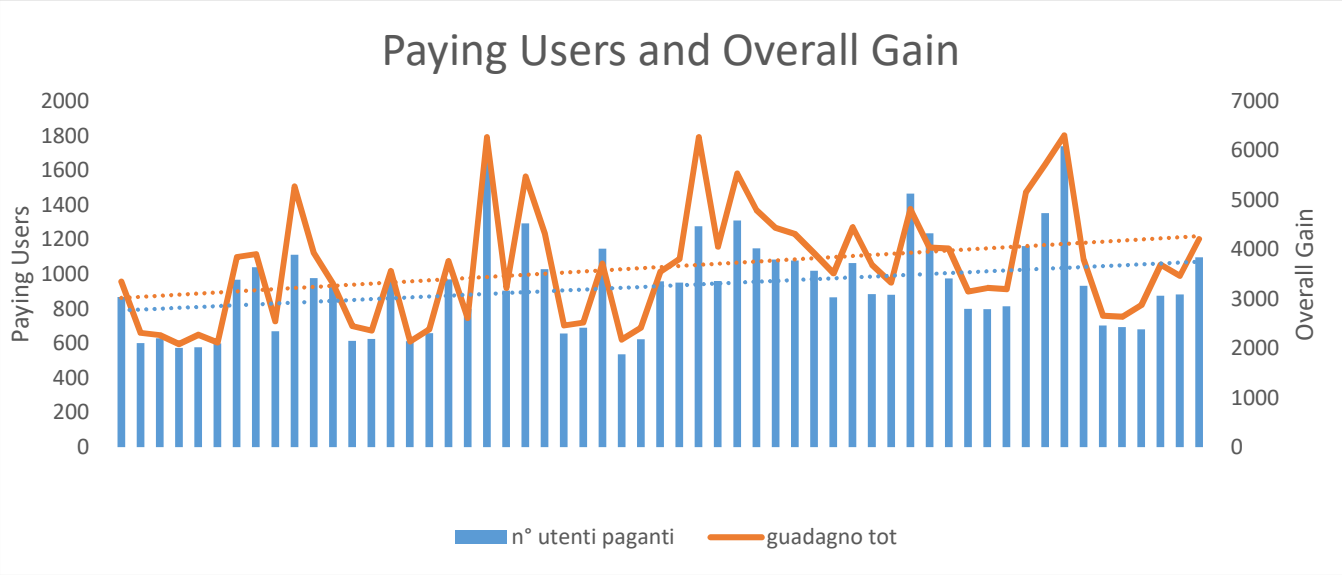


3 Purchase

I had to assume that I could use the database “purchase” too, otherwise if I use only the “retention” database, I would not have insights on the economic data.

3.1 How many users purchase IAPs and how much do they pay every day in total?

```
SELECT CAST(event_ts AS DATE) AS DAY,
COUNT(DISTINCT user_id) AS "n° Paying Users",
SUM(sku_price) AS "Overall Gain"
FROM purchases
GROUP BY DAY
ORDER BY DAY
```



For the reason explained in 1.1 I will remove from this data visualization the 28th of October.
The graph matches the intuition: the overall Gain is a function of the number of Paying Users.
Both trends seems to be in growth.

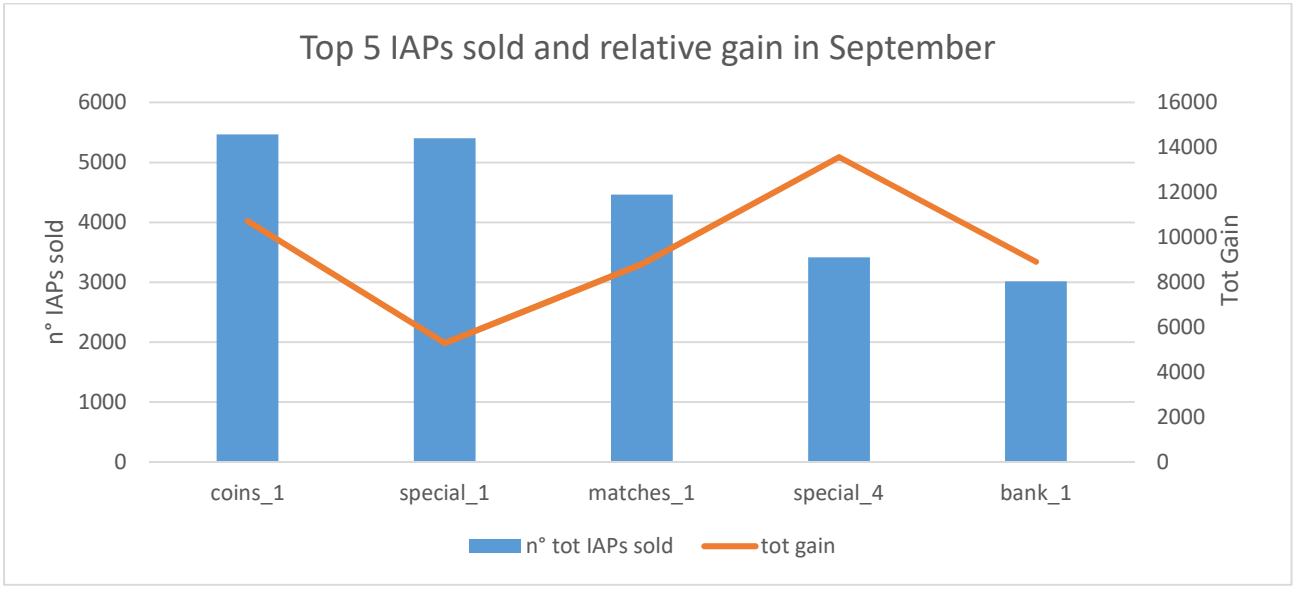
3.2 What are the top 5 most purchased IAPs in September and in October? How much did the users pay in total to buy these IAPs?

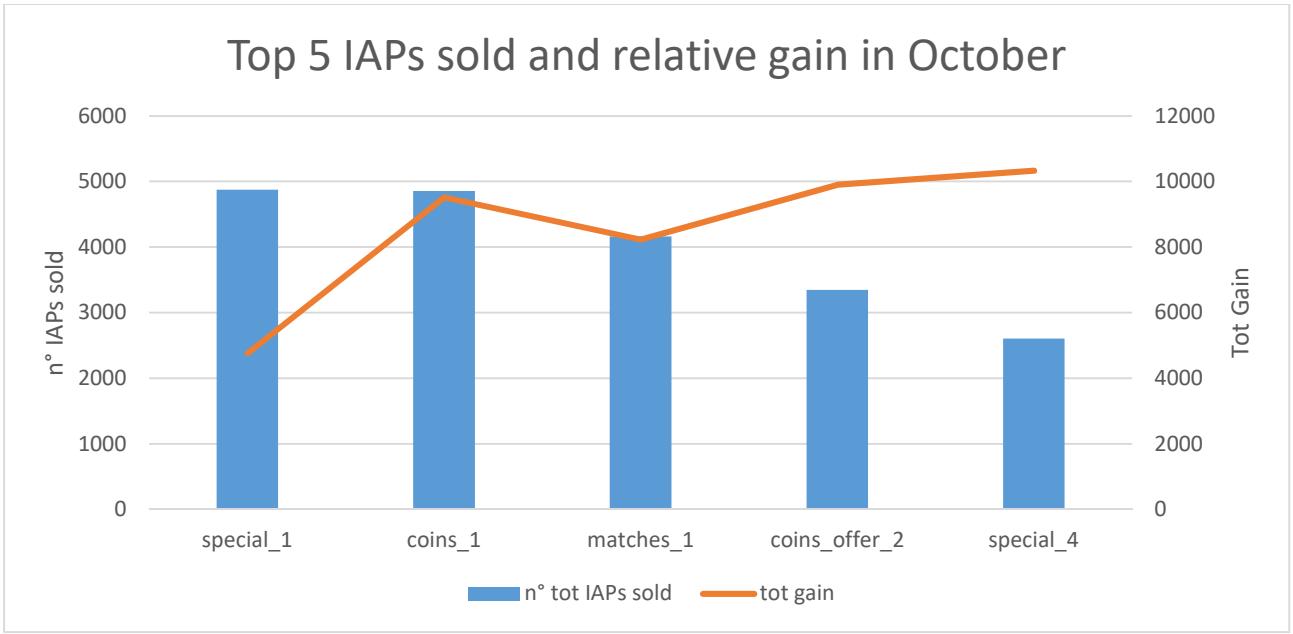
```
SELECT *
FROM
(
(SELECT EXTRACT(Month FROM event_ts) AS month, sku, COUNT(sku) AS "n° tot
acquisti", SUM(sku_price) AS "Guadagno tot"
FROM purchases
WHERE month = '9'
GROUP BY month, sku
ORDER BY COUNT(sku) DESC
LIMIT 5)
UNION
(SELECT EXTRACT(Month FROM event_ts) AS month, sku, COUNT(sku) AS "n° tot
acquisti", SUM(sku_price) AS "Guadagno tot"
FROM purchases
WHERE month = '10'
GROUP BY month, sku
```

```
ORDER BY COUNT(sku) DESC
LIMIT 5)
)
ORDER BY month
```

Cons_1, Special_1, Matches_1 are almost constant in number sold and total gain.

In both month “special_4” is a luxury item (high cost) that, despite its fewer tot sales, has the highest total gain of the Top 5 IAPs.

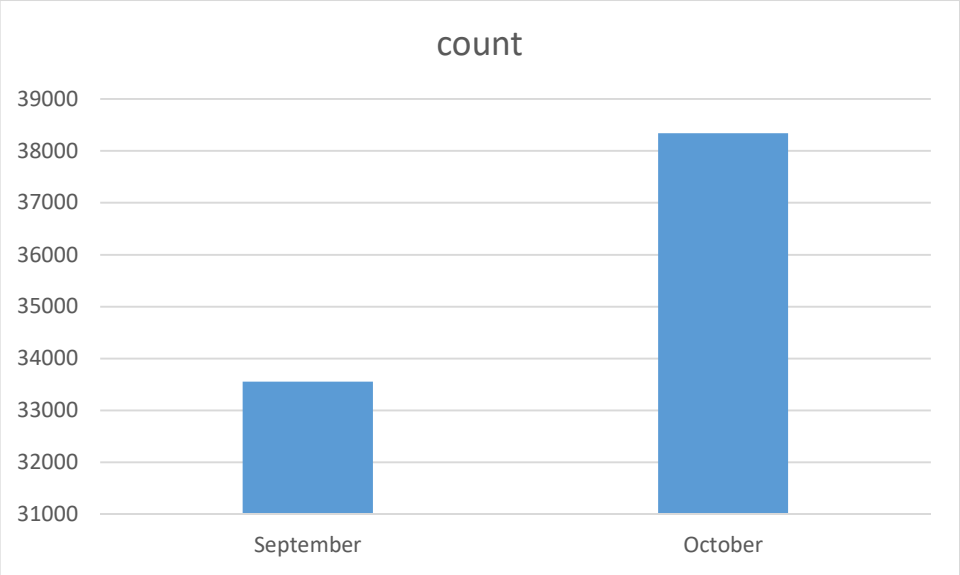




3.3 Are the users buying more IAPs in September or in October?

```
SELECT EXTRACT(Month FROM event_ts) AS month, COUNT(sku)
FROM purchases
GROUP BY month
ORDER BY COUNT(sku)
```

The Users bought +12,5% object in October than in September



3.4 How much do the users spend in total in their first week of playing the game?

```sql
SELECT purchases.user_id as player,
SUM(sku_price) AS "Tot Spent in first week"
FROM purchases
INNER JOIN (
     SELECT user_id, MIN(CAST(event_ts AS DATE)) AS first_join
     FROM purchases
     GROUP BY user_id
) AS tab
ON tab.user_id = purchases.user_id
WHERE CAST(event_ts AS DATE) BETWEEN first_join AND first_join+7
GROUP BY player
ORDER BY SUM(sku_price) DESC
```

In the average the users spent 5.27€ in their first week, with a standard deviation of 6.80€.
In total the amount gained by every users' first week is 75'613€.

We can try to reason about the relationship between the number of players that purchased at least one item (Paying Players ~14349) and the totality of players (Tot Players ~233301).
We can see that the ratio is around 1:16.

More interestingly, in this database we see confirmed a trend that had been foreseen on the exercise 1.3 that is typical of Mobile Games: the so called "Mobile Game Whales". In fact, the TOP 1% players (~2333) is paying around 48% of the total revenue (~36241€).