Common BI Metrics - Project

Delivery date: 18 Feb 2022

Part 1 - The database

The database

id	user_id	price	refunded_at	created_at
0	255	1.5	Ø	2015-10-04 14:39:46
1	448	0.5	Ø	2015-09-11 17:33:06
2	237	4.5	Ø	2015-08-17 23:13:32
3	468	3.5	Ø	2015-08-08 16:25:59
4	50	4.5	Ø	2015-08-18 02:36:17
5	86	1.5	2015-11-19 10:02:31	2015-11-19 05:16:54
6	384	4.5	Ø	2015-08-21 14:35:02
7	490	1.5	Ø	2015-10-08 01:09:32
8	454	1.5	Ø	2015-10-24 12:26:39
9	280	1.5	Ø	2015-08-24 15:19:00
10	109	3.5	Ø	2015-09-14 12:34:58
11	196	1.5	2015-11-20 16:10:43	2015-11-20 16:10:28
12	282	1.5	Ø	2015-10-17 23:39:39
13	487	4.5	Ø	2015-11-03 13:30:47
14	166	0.5	Ø	2015-09-21 03:33:22
15	239	1.5	Ø	2015-11-13 07:49:17
16	190	0.5	Ø	2015-10-26 04:25:58
17	369	3.5	Ø	2015-10-20 11:23:49
18	262	4.5	Ø	2015-09-12 09:25:49
19	77	0.5	Ø	2015-10-28 03:35:36
20	359	4.5	Ø	2015-09-13 17:48:34

The first 20 rows of the dataset

There are two tables:

purchases	
name	**type**
id	int
user_id	int

price	real
refunded_at	text
created_at	text

gameplays	
name	**type**
id	int
user_id	int
created_at	text
platform	text

Part 2 - Queries and results

Task description	SQL code

Average Revenue Per Purchasing User
 	
This metric shows if the average amount of money spent by purchasers is going up over time.
	select
	  date(created_at),
	  round(sum(price) /
	count(distinct(user_id)), 2) as arppu
	from purchases
	where refunded_at is null
	group by 1
	order by 1;

Finding average revenue per purchasing user using "with"	with daily_revenue as (
	  select
	  date(created_at) as dt,
	  round(sum(price), 2) as rev
	  from purchases
	  where refunded_at is null
	  group by 1

	) select * from daily_revenue order by dt;

Connecting ARPU by each date and joining with the daily average revenue per user	with daily_revenue as (
	  select
	  date(created_at) as dt,
	  round(sum(price), 2) as rev
	  from purchases
	  where refunded_at is null
	  group by 1
	),
	daily_players as (
	  select
	  date(created_at) as dt,
	  count(distinct user_id) as players
	  from gameplays
	  group by 1
	)
	select
	  daily_revenue.dt,
	  daily_revenue.rev / daily_players.players
	as arpu
	from daily_revenue
	  join daily_players using (dt);

This query returns a table of retention of all users. The retention is obtained by counting those who have been on the previous and the day after. It represents a comparison between the total users and retained users and is grouped by a date.	select
	  date(g1.created_at) as dt,
	  count(distinct g1.user_id) as total_users,
	  count(distinct g2.user_id) as
	retained_users
	from gameplays as g1
	  left join gameplays as g2 on
	  g1.user_id = g2.user_id
	  and date(g1.created_at) =
	date(datetime(g2.created_at, '-1 day'))
	group by 1
	order by 1
	limit 100;

Extrancing only count of retention from previous query	select
	  date(g1.created_at) as dt,
	  round(100 * count(distinct g2.user_id) /
	  count(distinct g1.user_id)) as retention

```
from gameplays as g1
  left join gameplays as g2 on
    g1.user_id = g2.user_id
    and date(g1.created_at) =
date(datetime(g2.created_at, '-1 day'))
group by 1
order by 1
```