Business Case

Streaming Platform Dataset Report

Marc Roig Lama

Contents

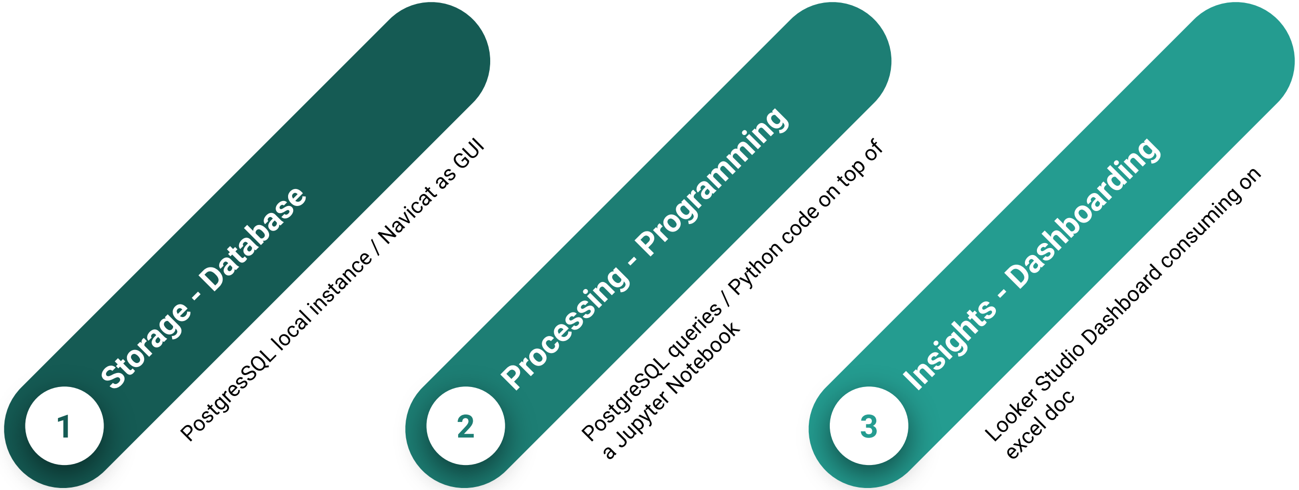
1. Context
2. Workflow
3. Storage - Database
4. Processing - Programming
5. Insights - Dashboarding

1. Context

1 sql database
to base the analysis

- Advertising-based Video on Demand Dataset
- 7 movies
- 5 thousand plays with revenue & viewtimes
- 1 week of data

2. Workflow



1 Storage - Database

PostgresSQL local instance / Navicat as GUI

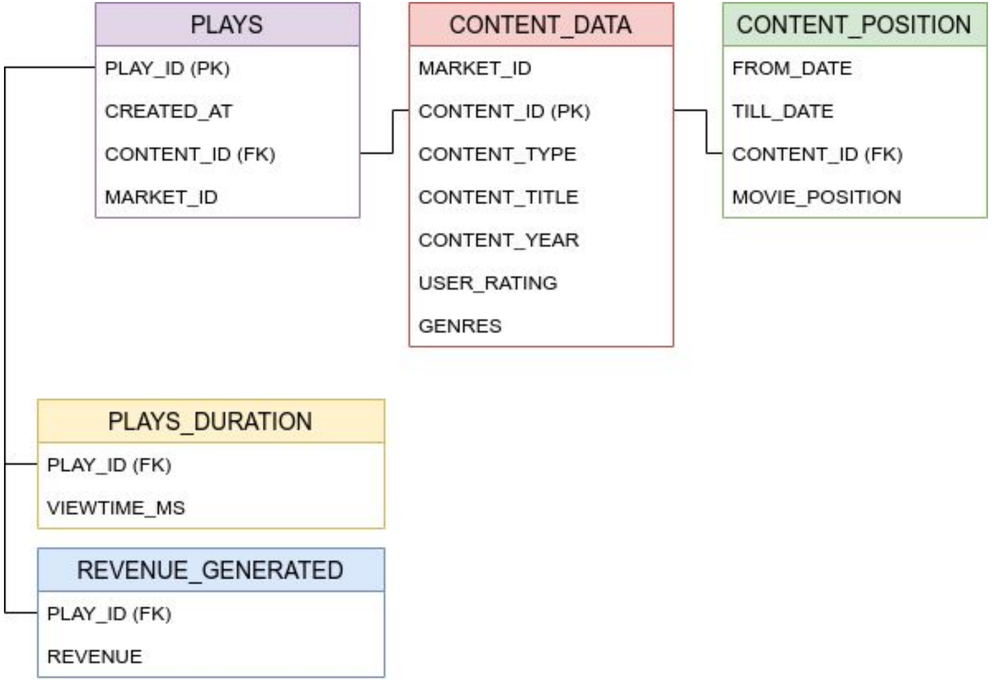
2 Processing - Programming

PostgreSQL queries / Python code on top of a Jupyter Notebook

3 Insights - Dashboarding

Looker Studio Dashboard consuming on excel doc

3. Storage - schema



PLAYS
- PLAY_ID (PK)
- CREATED_AT
- CONTENT_ID (FK)
- MARKET_ID

CONTENT_DATA
- MARKET_ID
- CONTENT_ID (PK)
- CONTENT_TYPE
- CONTENT_TITLE
- CONTENT_YEAR
- USER_RATING
- GENRES

CONTENT_POSITION
- FROM_DATE
- TILL_DATE
- CONTENT_ID (FK)
- MOVIE_POSITION

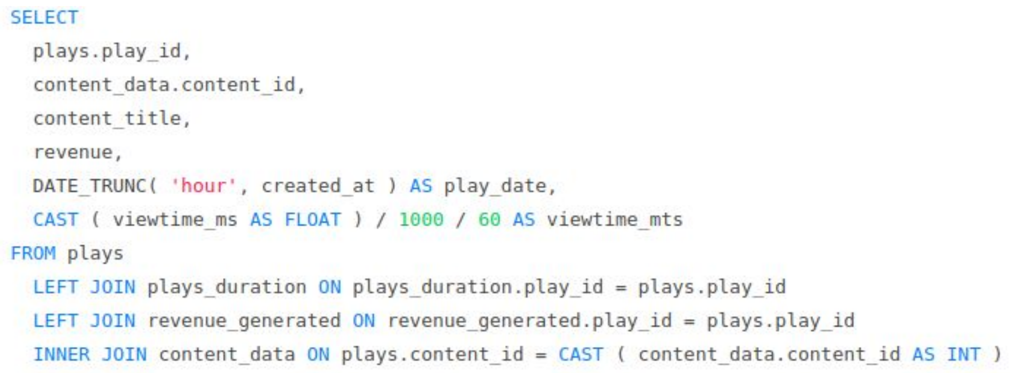
PLAYS_DURATION
- PLAY_ID (FK)
- VIEWTIME_MS

REVENUE_GENERATED
- PLAY_ID (FK)
- REVENUE

- 2 main tables:
  - Content Data & Plays
- I would create PK and FK
- Some columns haven’t the optimal type:
  - content_data.content_id is string
  - content_data.genres need to be normalized

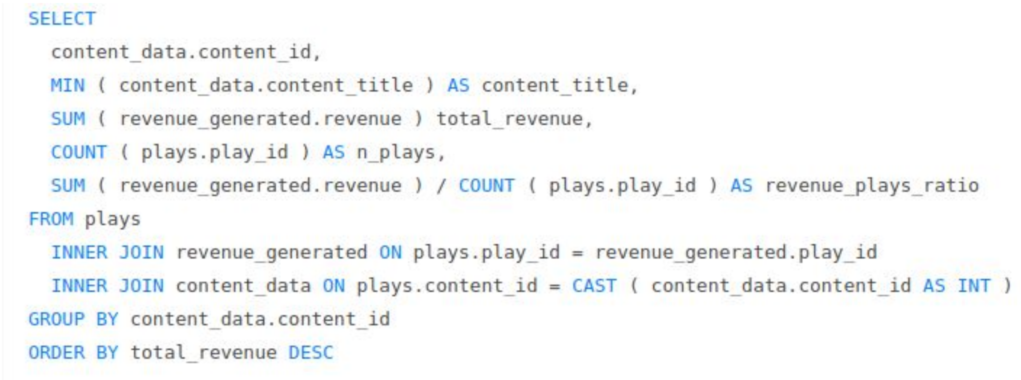
4. Processing - extract data

Plays Data



```
SELECT
  plays.play_id,
  content_data.content_id,
  content_title,
  revenue,
  DATE_TRUNC( 'hour', created_at ) AS play_date,
  CAST ( viewtime_ms AS FLOAT ) / 1000 / 60 AS viewtime_mts
FROM plays
  LEFT JOIN plays_duration ON plays_duration.play_id = plays.play_id
  LEFT JOIN revenue_generated ON revenue_generated.play_id = plays.play_id
  INNER JOIN content_data ON plays.content_id = CAST ( content_data.content_id AS INT )
```

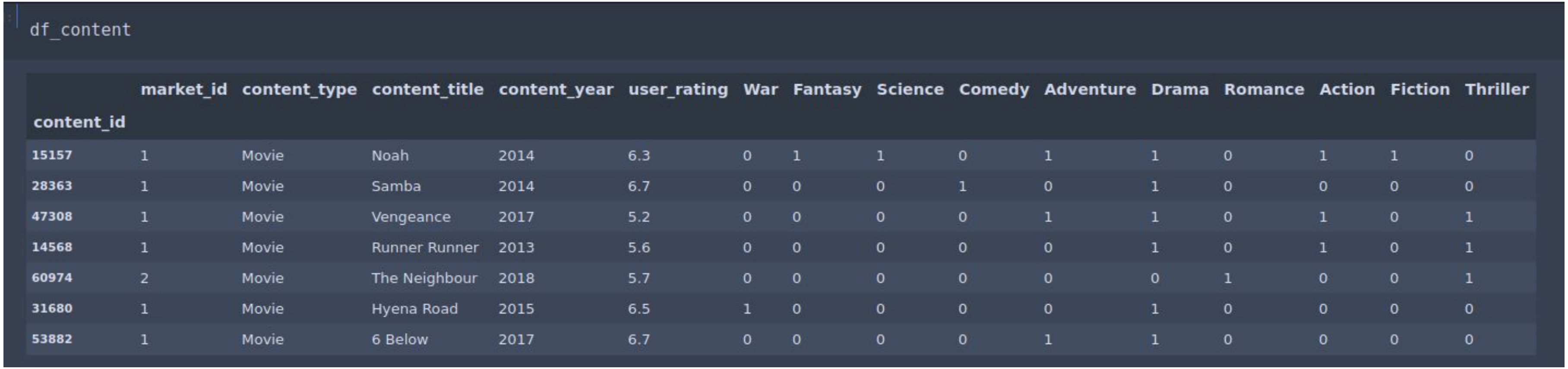
Movie Abstract



```
SELECT
  content_data.content_id,
  MIN ( content_data.content_title ) AS content_title,
  SUM ( revenue_generated.revenue ) total_revenue,
  COUNT ( plays.play_id ) AS n_plays,
  SUM ( revenue_generated.revenue ) / COUNT ( plays.play_id ) AS revenue_plays_ratio
FROM plays
  INNER JOIN revenue_generated ON plays.play_id = revenue_generated.play_id
  INNER JOIN content_data ON plays.content_id = CAST ( content_data.content_id AS INT )
GROUP BY content_data.content_id
ORDER BY total_revenue DESC
```

4. Processing - Content data

- Only 7 films

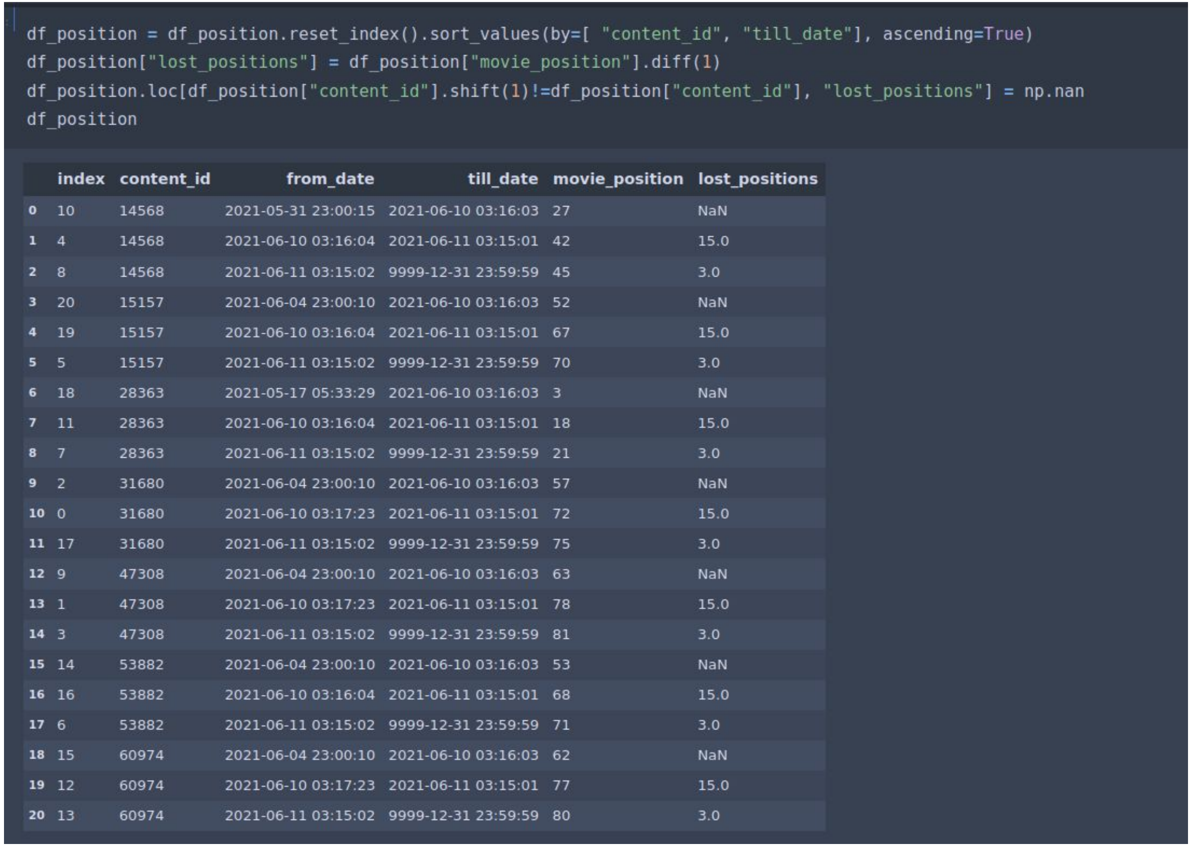


```
df_content
```

	market_id	content_type	content_title	content_year	user_rating	War	Fantasy	Science	Comedy	Adventure	Drama	Romance	Action	Fiction	Thriller
content_id															
15157	1	Movie	Noah	2014	6.3	0	1	1	0	1	1	0	1	1	0
28363	1	Movie	Samba	2014	6.7	0	0	0	1	0	1	0	0	0	0
47308	1	Movie	Vengeance	2017	5.2	0	0	0	0	1	1	0	1	0	1
14568	1	Movie	Runner Runner	2013	5.6	0	0	0	0	0	1	0	1	0	1
60974	2	Movie	The Neighbour	2018	5.7	0	0	0	0	0	0	1	0	0	1
31680	1	Movie	Hyena Road	2015	6.5	1	0	0	0	0	1	0	0	0	0
53882	1	Movie	6 Below	2017	6.7	0	0	0	0	1	1	0	0	0	0

4. Processing - Position data

- there is a pattern!
- Not a lot of changes (20 records)

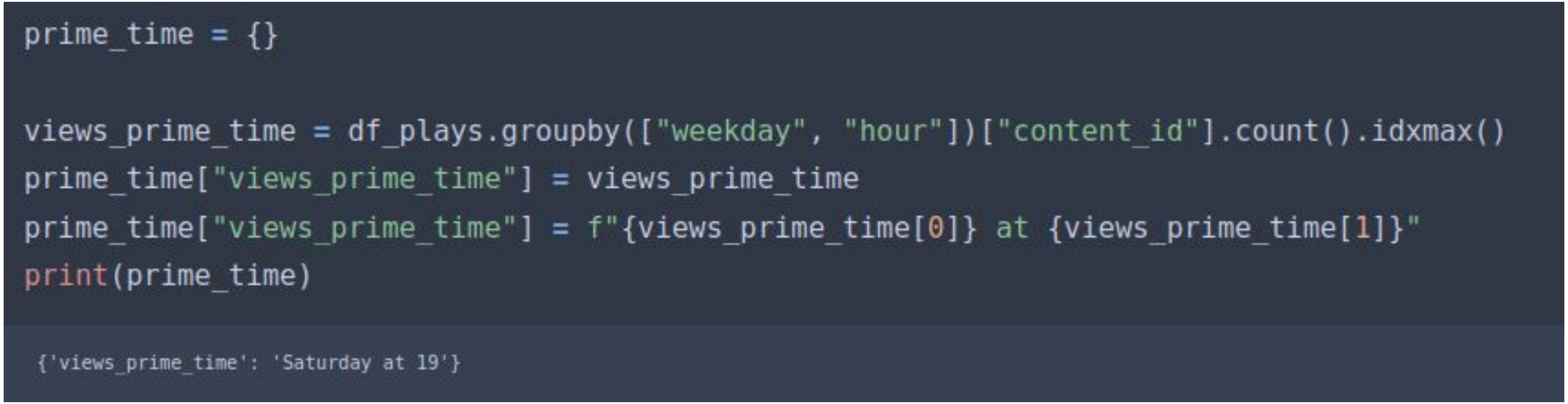


```
df_position = df_position.reset_index().sort_values(by=[ "content_id", "till_date"], ascending=True)
df_position["lost_positions"] = df_position["movie_position"].diff(1)
df_position.loc[df_position["content_id"].shift(1)!=df_position["content_id"], "lost_positions"] = np.nan
df_position
```

	index	content_id	from_date	till_date	movie_position	lost_positions
0	10	14568	2021-05-31 23:00:15	2021-06-10 03:16:03	27	NaN
1	4	14568	2021-06-10 03:16:04	2021-06-11 03:15:01	42	15.0
2	8	14568	2021-06-11 03:15:02	9999-12-31 23:59:59	45	3.0
3	20	15157	2021-06-04 23:00:10	2021-06-10 03:16:03	52	NaN
4	19	15157	2021-06-10 03:16:04	2021-06-11 03:15:01	67	15.0
5	5	15157	2021-06-11 03:15:02	9999-12-31 23:59:59	70	3.0
6	18	28363	2021-05-17 05:33:29	2021-06-10 03:16:03	3	NaN
7	11	28363	2021-06-10 03:16:04	2021-06-11 03:15:01	18	15.0
8	7	28363	2021-06-11 03:15:02	9999-12-31 23:59:59	21	3.0
9	2	31680	2021-06-04 23:00:10	2021-06-10 03:16:03	57	NaN
10	0	31680	2021-06-10 03:17:23	2021-06-11 03:15:01	72	15.0
11	17	31680	2021-06-11 03:15:02	9999-12-31 23:59:59	75	3.0
12	9	47308	2021-06-04 23:00:10	2021-06-10 03:16:03	63	NaN
13	1	47308	2021-06-10 03:17:23	2021-06-11 03:15:01	78	15.0
14	3	47308	2021-06-11 03:15:02	9999-12-31 23:59:59	81	3.0
15	14	53882	2021-06-04 23:00:10	2021-06-10 03:16:03	53	NaN
16	16	53882	2021-06-10 03:16:04	2021-06-11 03:15:01	68	15.0
17	6	53882	2021-06-11 03:15:02	9999-12-31 23:59:59	71	3.0
18	15	60974	2021-06-04 23:00:10	2021-06-10 03:16:03	62	NaN
19	12	60974	2021-06-10 03:17:23	2021-06-11 03:15:01	77	15.0
20	13	60974	2021-06-11 03:15:02	9999-12-31 23:59:59	80	3.0

4. Processing - Prime Time

Day of the week and hour with more views/plays



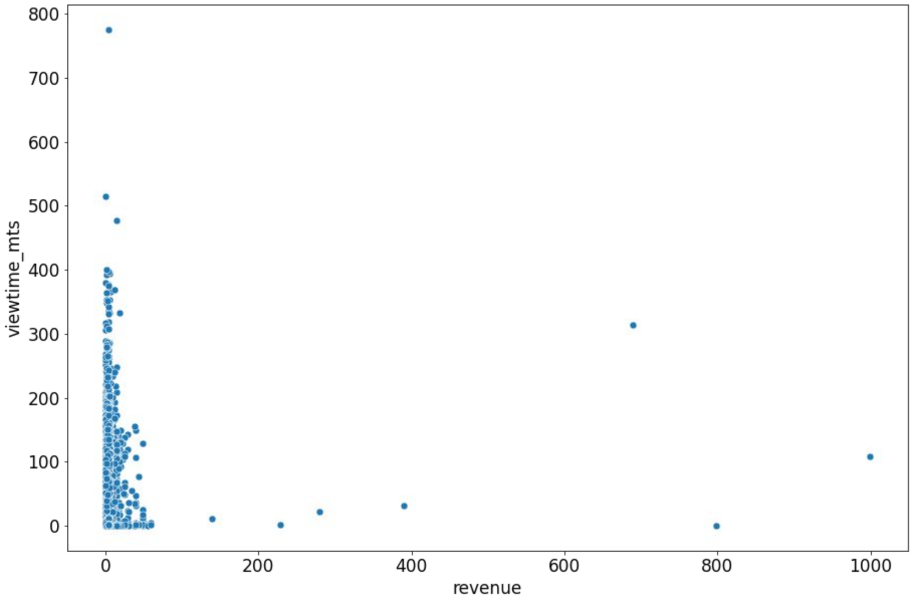
```
prime_time = {}

views_prime_time = df_plays.groupby(["weekday", "hour"])["content_id"].count().idxmax()
prime_time["views_prime_time"] = views_prime_time
prime_time["views_prime_time"] = f"{views_prime_time[0]} at {views_prime_time[1]}"
print(prime_time)
```

```
{'views_prime_time': 'Saturday at 19'}
```

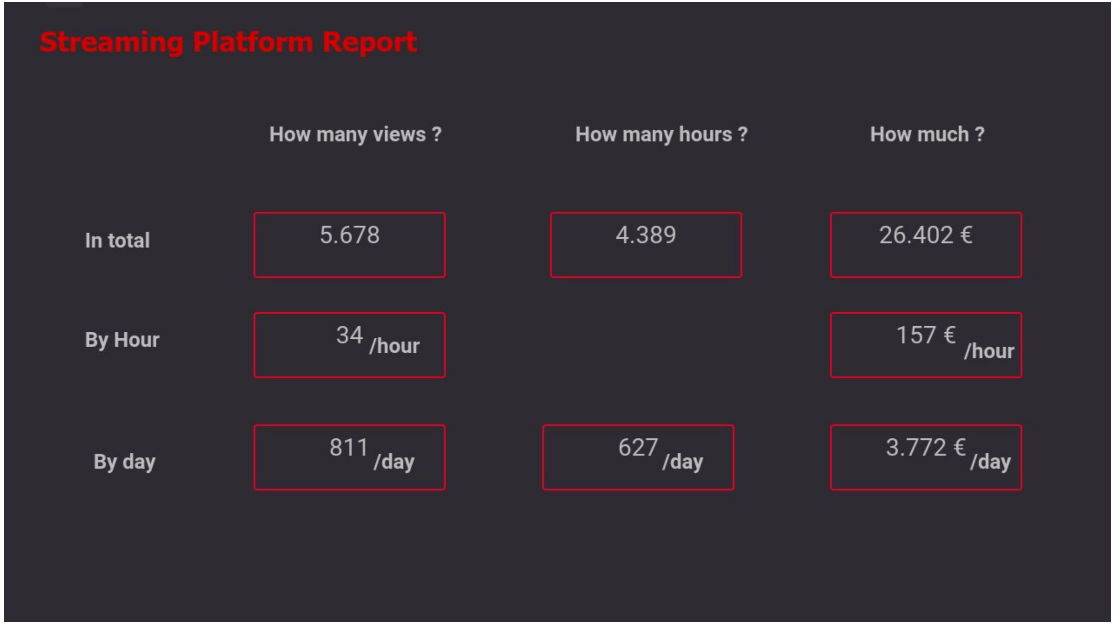
4. Processing - Revenue VS viewtime

Hypothesis: if revenue is based on commercials and commercials appears every N minutes, **view time should be highly correlated with revenue**.



- Doesn’t seem to be correlated.
- After removing some outliers, there is no correlation at all

5. Insights - Dashboarding



Streaming Platform Report

	How many views ?	How many hours ?	How much ?
In total	5.678	4.389	26.402 €
By Hour	34 /hour		157 € /hour
By day	811 /day	627 /day	3.772 € /day

Streaming Platform Dataset Analysis