Aim of business:

The business is a streaming platform called “***streamio***”. The company stores information about streamers, viewers, games, keeps track of the donations and the streams details. The company has a simple runnable program which will allows the user to be able to do a few operations through a simple interface.

These operations include **viewing the tables**, **viewing specific records in tables**, **adding new records to the tables**; hence registering as streamers, viewers, registering games to stream and keeping records of streams.

There are 2 more operations which are **donating to a streamer** and **executing more MySQL commands**:

- By **donating to a streamer**, the user, who should be a registered viewer, can choose a streamer and input the amount to be donated, where the code will check the database if there are enough funds in the viewer’s wallet to commence the operation. This adds a new record in the *donation* table.
- By **executing more MySQL commands**, the user can input any MySQL command in the correct syntax and the code will be able to execute it.

Database design:

The database consists of 5 entities which include *streamer*, *viewer*, *game, donation, stream* and 2 relations which include *donate* and *streaming*.

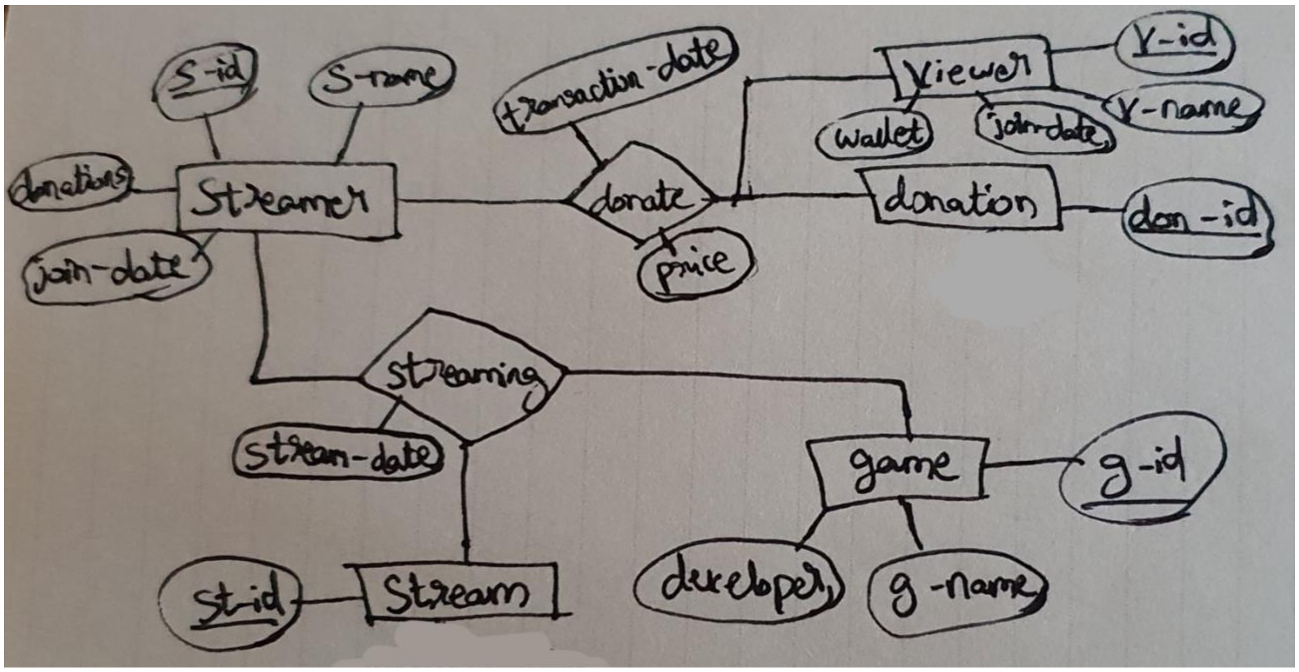
Donate is a ternary relationship between streamer, viewer, and donation while *streaming* Is another ternary relation between *streamer*, *game,* and *stream*.

The relationships *donate* and *streaming* are both implicitly implemented as they fit into the tables and do not cause any anomalies since *donation* and *stream* tuples cannot be created without *donate* and *streaming* respectively.

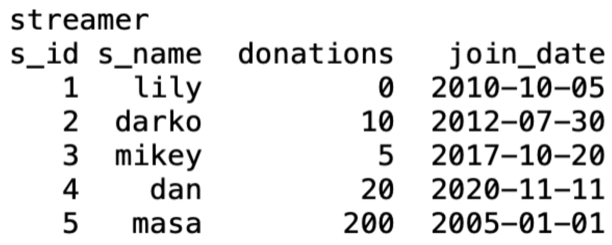
Tables:

- ***Streamer*** : stores streamer id, name, total donations received and the date they joined.
  (s_id, s_name, donations, join_date)
- ***Viewer*** : stores viewer id, name, date they joined and the funds in their wallet.
  (v_id, v_name, join_date, wallet)
- ***Game*** : stores game id, name and developer.
  (g_id, g_name, developer)
- ***Donation*** : stores donation id and since it implicitly implements *donate* it also contains the amount paid hence price, transaction date and the ids of the streamer and the viewer involved.
  (don_id, s_id, v_id, price, transaction_date)
- ***Stream*** : stores stream id and since it implicitly implements *streaming* it also contains the stream date and the ids of the streamer and game involved.
  (st_id, s_id, g_id, stream_date)

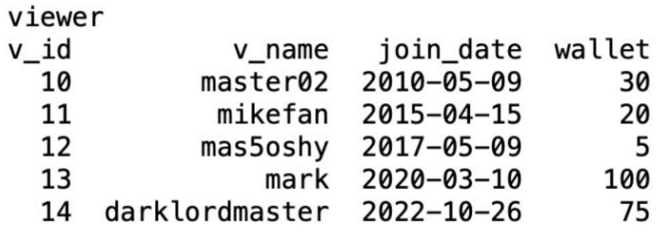
ER Diagram:



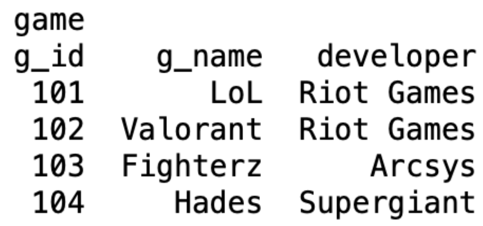
Snapshots:



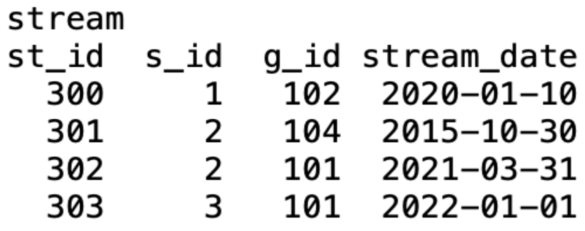
```
streamer
s_id s_name  donations   join_date
   1   lily          0  2010-10-05
   2  darko         10  2012-07-30
   3  mikey          5  2017-10-20
   4    dan         20  2020-11-11
   5   masa        200  2005-01-01
```



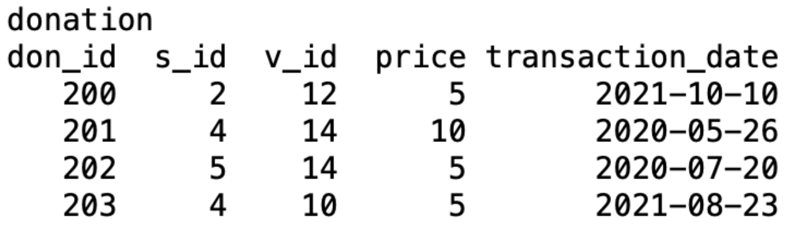
```
viewer
v_id         v_name   join_date  wallet
  10       master02  2010-05-09      30
  11        mikefan  2015-04-15      20
  12       mas5oshy  2017-05-09       5
  13           mark  2020-03-10     100
  14  darklordmaster  2022-10-26      75
```



```
game
g_id    g_name   developer
 101       LoL  Riot Games
 102  Valorant  Riot Games
 103  Fighterz      Arcsys
 104     Hades  Supergiant
```



```
stream
st_id  s_id  g_id stream_date
  300     1   102  2020-01-10
  301     2   104  2015-10-30
  302     2   101  2021-03-31
  303     3   101  2022-01-01
```



```
donation
don_id  s_id  v_id  price transaction_date
   200     2    12      5       2021-10-10
   201     4    14     10       2020-05-26
   202     5    14      5       2020-07-20
   203     4    10      5       2021-08-23
```

Keys:

- ***Streamer*** : s_id is the primary key since it is unique for every streamer.
- ***Viewer*** : v_id is the primary key since it is unique for every viewer.
- ***Game*** : g_id is the primary key since it is unique for every game.
- ***Donation*** : don_id is the primary key since it is unique for every donation while s_id and v_id are both foreign keys from streamer and viewer tables respectively.
- ***Stream*** : st_id is the primary key since it is unique for every stream while s_id and g_id are both foreign keys from streamer and game tables respectively.

Cardinalities:

- ***Donate*** : many to many to many since streamers can have multiple donations from multiple viewers and viewers can donate to multiple streamers. In addition, a viewer can donate to the same streamer multiple times.
- ***Streaming*** : many to many to many since streamers can stream multiple games and games can be streamed by multiple streamers. Moreover, the same combination of streamer and game can occur multiple times.

Normalization:

- ***1NF*** : all the tables in the database are already in ***1NF*** since all the columns have atomic values hence no composite nor multivalued columns. For example: *Streamer* has an atomic value for total of donations received, *viewer* has atomic value for wallet, game has a singular developer.

- ***2NF*** : all the tables in the database are already in ***2NF*** since they are in ***1NF*** and none of the tables have partial dependencies hence all the columns in the table rely on the same column which is the primary key. For instance: in *streamer* all the columns depend on the primary key s_id and this applies to all the other tables with their respective primary key and respective columns.

- ***3NF*** : all the tables in the database are already in ***3NF*** since they are in ***2NF*** and there is no transitive dependencies in any of the tables since all the columns depend on a singular column as mentioned in ***2NF***.

- ***4NF*** : all the tables in the database are already in ***4NF*** since they are in all the previous normalization forms and since the tables do not contain multivalued dependencies. For example: all the columns in *stream* are significant and impact each other since all the data is connected.