dmin@bithub-bot:~/Bot/app$ cat ^C

admin@bithub-bot:~/Bot/app$ cat db\_query.py

import db\_models

from helper import insert\_time

import text\_file

from datetime import datetime, timedelta

class QueryDb:

@classmethod

def save\_user\_state(cls, uid, f\_name, state, prev):

try:

user = db\_models.Users.create(uid=uid, f\_name=f\_name, state=state, prev=prev)

db\_models.Profit.create(uid=uid)

except Exception:

db\_models.db.rollback()

user = db\_models.Users.get(db\_models.Users.uid == uid)

user.state = state

user.prev = prev

user.save()

return user

@classmethod

def get\_user(cls, uid):

try:

user = db\_models.Users.get(db\_models.Users.uid == uid)

except db\_models.Users.DoesNotExist:

return None

return user

@classmethod

def save\_sub(cls, user, days=None):

sub = user.sub

if sub:

sub[0].date = insert\_time(sub, days)

sub[0].group = '👻 Новичок'

sub[0].save()

return sub[0]

return db\_models.Subscribers.create(uid=user.uid, group=user.prev, date=insert\_time(sub, days))

@classmethod

def save\_admin\_sub(cls, number, group, days):

try:

sub = db\_models.Subscribers.select().where(db\_models.Subscribers.id == number)

except ValueError:

return None

else:

if sub:

sub[0].id = number

if sub[0].group != text\_file.group.get(group):

sub[0].group = text\_file.group.get(group)

if days:

sub[0].date = datetime.now() + timedelta(days=days)

else:

sub[0].date = None

sub[0].perm = True

else:

sub[0].group = text\_file.group.get(group)

if days:

sub[0].date = insert\_time(sub, days)

else:

sub[0].date = None

sub[0].perm = True

sub[0].save()

return sub[0]

return None

@classmethod

def check\_ref(cls, uid):

return db\_models.Referrals.select().where(db\_models.Referrals.uid == uid)

@classmethod

def add\_money(cls, ref, money):

try:

user = db\_models.Profit.get(db\_models.Profit.uid == ref)

except db\_models.Profit.DoesNotExist:

return None

else:

user.count += money

user.save()

return user

@classmethod

def get\_money(cls, uid, money):

user = cls.get\_user(uid)

profit = user.profit

if profit:

if profit[0].count < 0.01 or (profit[0].count - money) < 0:

return None

profit[0].count -= money

profit[0].save()

return profit[0]

return None

@staticmethod

def \_check\_user(uid):

user = db\_models.Users.select().where(db\_models.Users.uid == uid)

if user:

return True

return False

@classmethod

def add\_referral(cls, user, referral):

if cls.\_check\_user(user.chat.id):

return False

cls.save\_user\_state(user.chat.id, user.chat.first\_name, user.text, None)

db\_models.Referrals.create(referral=int(referral), uid=user.chat.id)

return True

@classmethod

def permissions(cls, uid, delete=False):

user = cls.get\_user(uid)

if user is not None:

if delete:

user.admin = False

else:

user.admin = True

user.save()

return True

return False

@classmethod

def get\_all\_admins(cls):

return db\_models.Users.select().where(db\_models.Users.admin == True)

@classmethod

def save\_wiki(cls, name, link, description):

try:

db\_models.Wiki.create(name=name, link=link, description=description)

except Exception:

return False

return True

@classmethod

def get\_wiki(cls):

return db\_models.Wiki.select()

@classmethod

def delete\_wiki(cls, name):

return db\_models.Wiki.delete().where(db\_models.Wiki.name == name).execute()

@classmethod

def mass\_update(cls, group, days):

if group:

update = db\_models.Subscribers.update(date=db\_models.Subscribers.date + timedelta(days=days)).where(

(db\_models.Subscribers.date != None) & (db\_models.Subscribers.group << text\_file.groups.get(group))

)

return update.execute()

update = db\_models.Subscribers.update(date=db\_models.Subscribers.date + timedelta(days=days)).where(

(db\_models.Subscribers.date != None) & (db\_models.Subscribers.group == '👻 Новичок')

)

return update.execute()

@classmethod

def statistics(cls):

total = db\_models.Users.select().count()

noobs = db\_models.Subscribers.select().where(db\_models.Subscribers.group == '👻 Новичок').count()

traders = db\_models.Subscribers.select().where(

(db\_models.Subscribers.group == '💸 Трейдер') & ((db\_models.Subscribers.date.is\_null(False)) |

(db\_models.Subscribers.perm == True))).count()

vips = db\_models.Subscribers.select().where(

(db\_models.Subscribers.group == '💵 VIP') & ((db\_models.Subscribers.date.is\_null(False)) |

(db\_models.Subscribers.perm == True))).count()

platinums = db\_models.Subscribers.select().where(

(db\_models.Subscribers.group == '💰 Platinum') & ((db\_models.Subscribers.date.is\_null(False)) |

(db\_models.Subscribers.perm == True))).count()

diamonds = db\_models.Subscribers.select().where(

(db\_models.Subscribers.group == '💎 Diamond')& ((db\_models.Subscribers.date.is\_null(False)) |

(db\_models.Subscribers.perm == True))).count()

current\_w = datetime.now() - timedelta(days=7)

current\_m = datetime.now() - timedelta(days=31)

week = db\_models.Users.select().where(db\_models.Users.date\_reg > current\_w).count()

month = db\_models.Users.select().where(db\_models.Users.date\_reg > current\_m).count()

referrals = {ref.referral.uid for ref in db\_models.Referrals.select()}

return total, noobs, traders, vips, platinums, diamonds, week, month, len(referrals)

@classmethod

def get\_users\_id(cls, photo=False):

if photo:

return [user.uid.uid for user in db\_models.Subscribers.select().where(

db\_models.Subscribers.group << ['💵 VIP', '💰 Platinum', '💎 Diamond'])]

count = db\_models.Culc.get(db\_models.Culc.id == 1)

group = []

if count.count in [1,4]:

group.append('👻 Новичок')

if count.count in [1,3,5]:

group.append('💸 Трейдер')

if count.count == 6:

count.count = 1

count.save()

else:

count.count += 1

count.save()

group += ['💵 VIP', '💰 Platinum', '💎 Diamond']

users = [user.uid.uid for user in db\_models.Subscribers.select().where(db\_models.Subscribers.group << group)]

return users

admin@bithub-bot:~/Bot/app$ ^C

admin@bithub-bot:~/Bot/app$