

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

# Кафедра системного програмування та спеціалізованих комп’ютерних систем

**Лабораторна робота №3**

з дисципліни

**«Бази даних і засоби управління»**

Виконав: студент ІII курсу

ФПМ групи КВ-83

Пащенко Антон

Київ – 2020

**Засоби оптимізації роботи СУБД PostgreSQL**

*Метою роботи* є здобуття практичних навичок використання засобів

оптимізації СУБД PostgreSQL.

*Завдання* роботи полягає у наступному:

1. Перетворити модуль “Модель” з шаблону MVC лабораторної роботи №2

у вигляд об’єктно-реляційної проекції (ORM).

1. Створити та проаналізувати різні типи індексів у PostgreSQL.
2. Розробити тригер бази даних PostgreSQL.

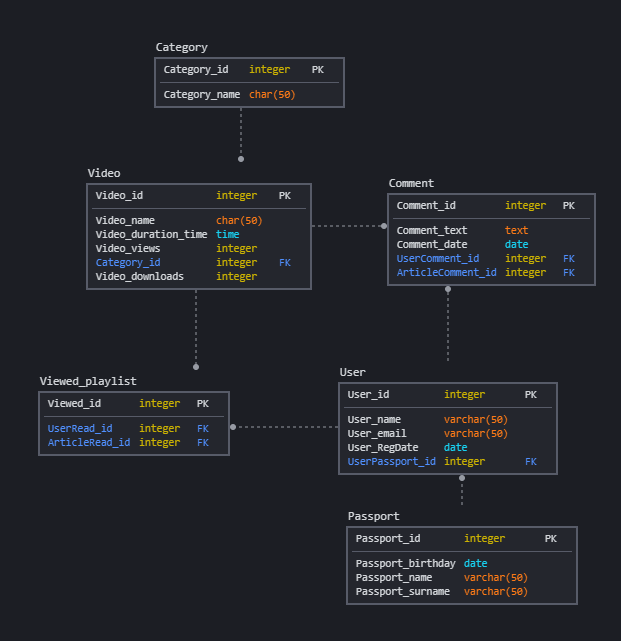
**Варіант 19**

У другому завданні проаналізувати індекси BTREE,BRIN.

Умова для тригера – before delete,insert

**Завдання 1**

Логічна схема бази даних “Відеохостінг”



Для перетворення модулю “Model” програми, створеної в 2 лабораторній роботі, у вигляд об’єктно-реляційної моделі використовую бібліотеку sqlalchemy.

Зобразимо сутнісні класи програми:

from sqlalchemy.ext.declarative import declarative\_base

from sqlalchemy import Column, Integer, String, Date,Text,DateTime,ForeignKey

from sqlalchemy.orm import relationship

import datetime

Base = declarative\_base()

class Category(Base):

\_\_tablename\_\_ = 'category'

id = Column('category\_id',Integer, primary\_key=True)

category\_name = Column(String(50))

def \_\_init\_\_(self,data):

self.category\_name = data['category\_name']

def change\_data(self,data):

for key,value in data.items():

setattr(self,key,value)

def to\_tuple(self):

return (self.id,self.category\_name)

def get\_column\_names():

return ['category\_id','category\_name']

class Comentary(Base):

\_\_tablename\_\_ = 'commentaries'

id = Column('comment\_id',Integer, primary\_key=True)

comment\_text = Column(Text)

comment\_to\_video = Column(Integer,ForeignKey('video.video\_id'))

comment\_from\_user = Column(Integer,ForeignKey('users.user\_id'))

video = relationship("Video",backref="commentaries")

user = relationship("User",backref="commentaries")

def \_\_init\_\_(self,data):

self.comment\_text = data['comment\_text']

self.comment\_to\_video = int(data['comment\_to\_video'])

self.comment\_from\_user = int(data['comment\_from\_user'])

def change\_data(self,data):

for key,value in data.items():

if key == 'comment\_to\_video':

self.comment\_to\_video = int(value)

elif key == 'comment\_from\_user':

self.comment\_from\_user = int(value)

else:

setattr(self,key,value)

def to\_tuple(self):

return (self.id,self.comment\_text,self.comment\_to\_video,self.comment\_from\_user)

def get\_column\_names():

return ['comment\_id','comment\_text','comment\_to\_video','comment\_from\_user']

class Passport(Base):

\_\_tablename\_\_ = 'passport'

id = Column('passport\_id',Integer, primary\_key=True)

passport\_name = Column(String(50))

passport\_surname = Column(String(50))

passport\_date = Column(Date)

def \_\_init\_\_(self,data):

self.passport\_name = data['passport\_name']

self.passport\_surname = data['passport\_surname']

self.passport\_date = datetime.datetime.strptime(data['passport\_date'], '%Y-%m-%d').date()

def change\_data(self,data):

for key,value in data.items():

if key == 'passport\_date':

self.passport\_date = datetime.datetime.strptime(value, '%Y-%m-%d').date()

else:

setattr(self,key,value)

def to\_tuple(self):

return (self.id,self.passport\_name,self.passport\_surname,self.passport\_date)

def get\_column\_names():

return ['passport\_id','passport\_name','passport\_surname','passport\_date']

class User(Base):

\_\_tablename\_\_ = 'users'

id = Column('user\_id',Integer, primary\_key=True)

user\_name = Column(String(50))

user\_email = Column(String(50))

user\_regdate = Column(Date)

user\_passport\_id = Column(Integer, ForeignKey('passport.passport\_id', ondelete='CASCADE'))

passport = relationship("Passport",cascade="all,delete",backref="user",passive\_deletes=True)

def \_\_init\_\_(self,data):

self.user\_name = data['user\_name']

self.user\_name = data['user\_name']

self.user\_email = data['user\_email']

self.user\_regdate = datetime.datetime.strptime(data['user\_regdate'], '%Y-%m-%d').date()

self.user\_passport\_id = int(data['user\_passport\_id'])

def change\_data(self,data):

for key,value in data.items():

if key == 'user\_regdate':

self.user\_regdate = datetime.datetime.strptime(value, '%Y-%m-%d').date()

elif key == 'user\_passport\_id':

self.user\_passport\_id = int(value)

else:

setattr(self,key,value)

def to\_tuple(self):

return (self.id,self.user\_name,self.user\_email,self.user\_regdate,self.user\_passport\_id)

def get\_column\_names():

return ['user\_id','user\_name','user\_email','user\_regdate','user\_passport\_id']

class Video(Base):

\_\_tablename\_\_ = 'video'

id = Column('video\_id',Integer, primary\_key=True)

video\_name = Column(String(50))

video\_duration\_time = Column(DateTime(timezone=False))

video\_views = Column(Integer)

video\_downloads = Column(Integer)

video\_category = Column(Integer,ForeignKey('category.category\_id'))

category = relationship("Category",backref="video")

def \_\_init\_\_(self,data):

self.video\_name = data['video\_name']

self.video\_duration\_time = datetime.datetime.strptime(data['video\_duration\_time'], '%H:%M:%S').time()

self.video\_views = int(data['video\_views'])

self.video\_downloads = int(data['video\_downloads'])

self.video\_category = int(data['video\_category'])

def change\_data(self,data):

for key,value in data.items():

if key == 'video\_duration\_time':

self.video\_duration\_time = datetime.datetime.strptime(value, '%H:%M:%S').time()

elif key == 'video\_views':

self.video\_views = int(value)

elif key == 'video\_downloads':

self.video\_downloads = int(value)

elif key == 'video\_category':

self.video\_category = int(value)

else:

setattr(self,key,value)

def to\_tuple(self):

return (self.id,self.video\_name,self.video\_duration\_time,self.video\_views,self.video\_downloads,self.video\_category)

def get\_column\_names():

return ['video\_id','video\_name','video\_duration\_time','video\_views','video\_downloads','video\_category']

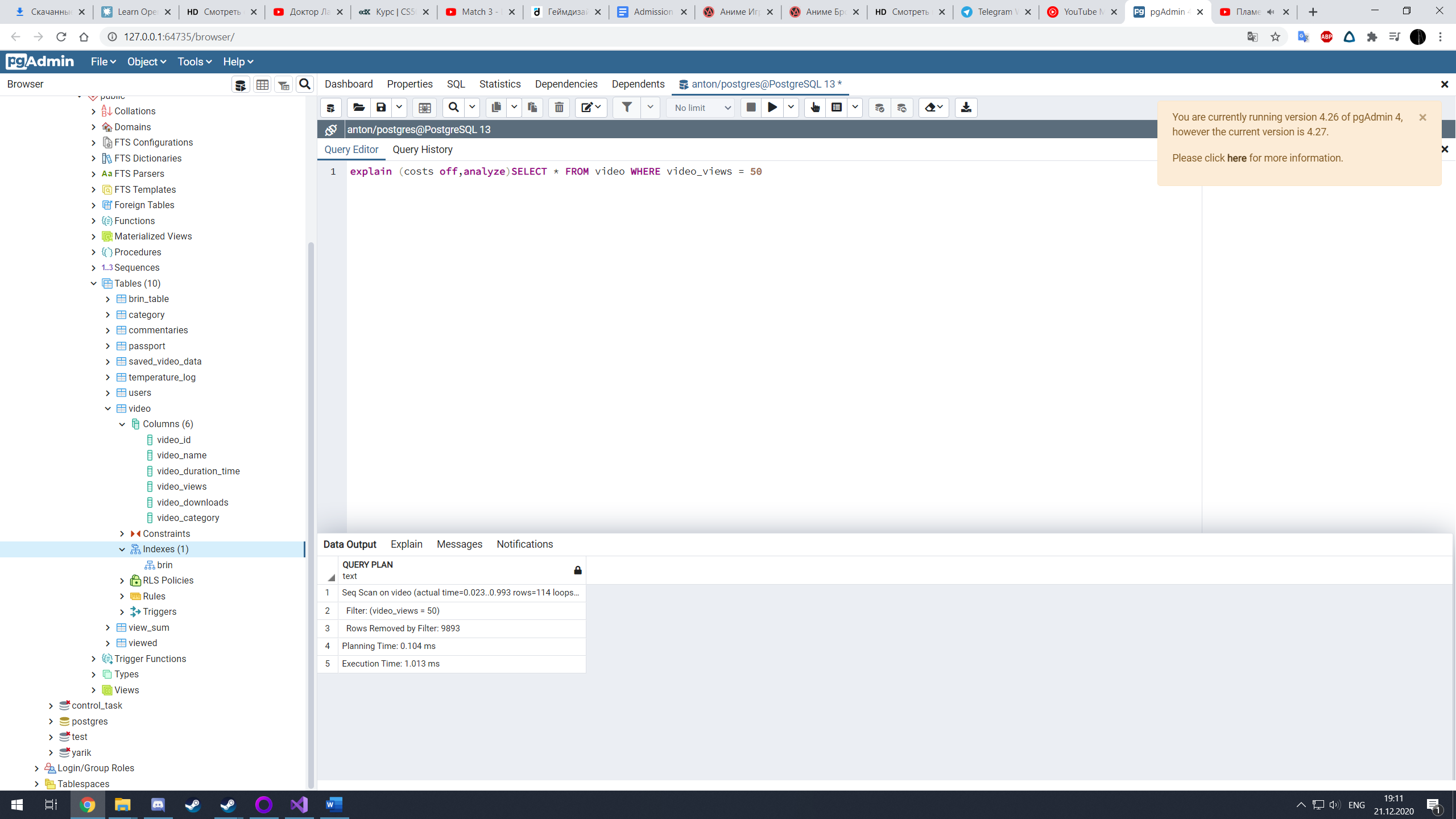
**Завдання 2**

**Створення та аналіз індекса BTREE**

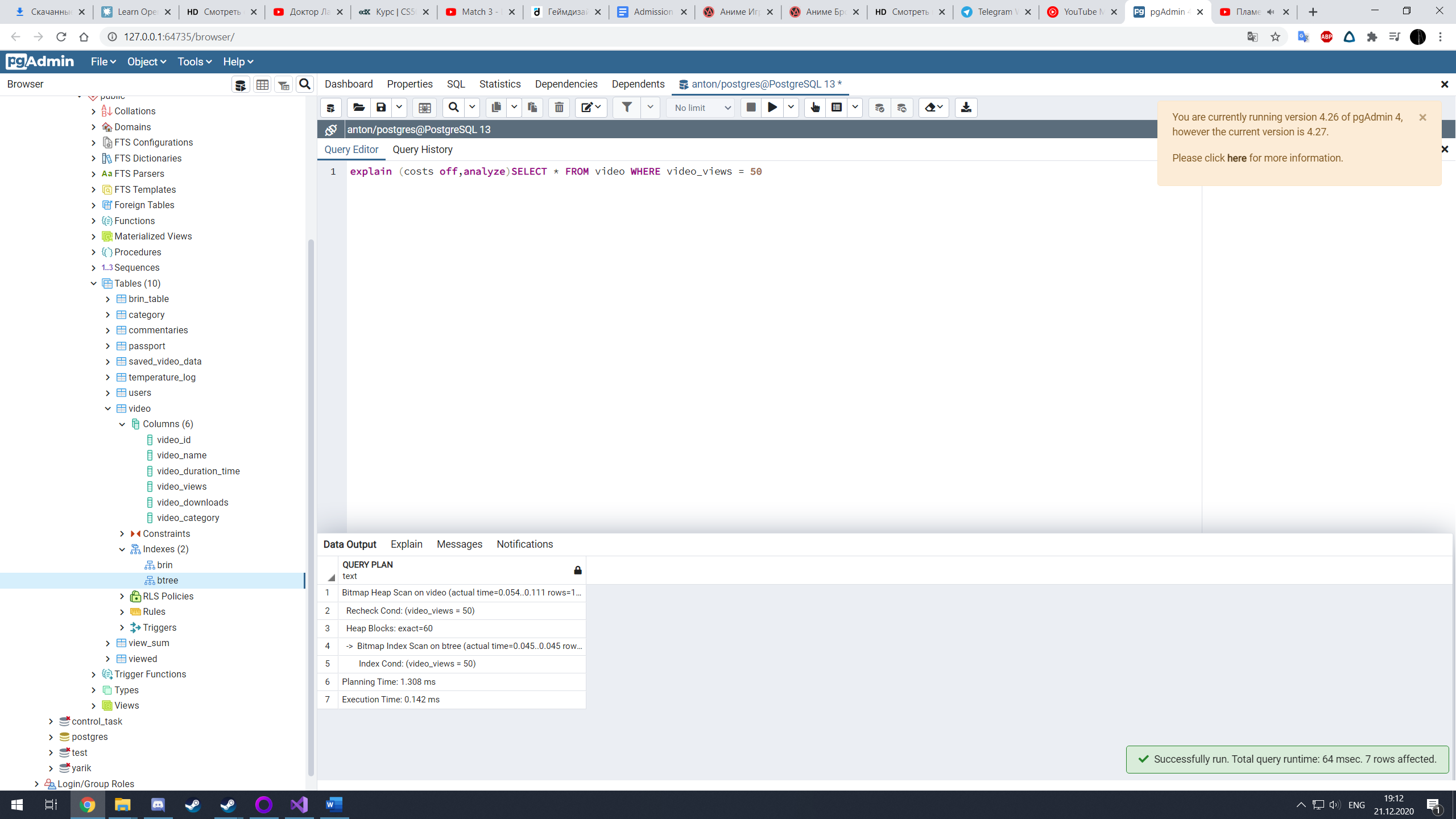
Для дослідження індексу була використана таблиця video та поле . Вони проіндексовані як BTREE. У таблицю було занесено 100000 записів. Виконаний запит.

SELECT \* FROM video WHERE video\_views = 50

До iндексування.



Пiсля iндексування.



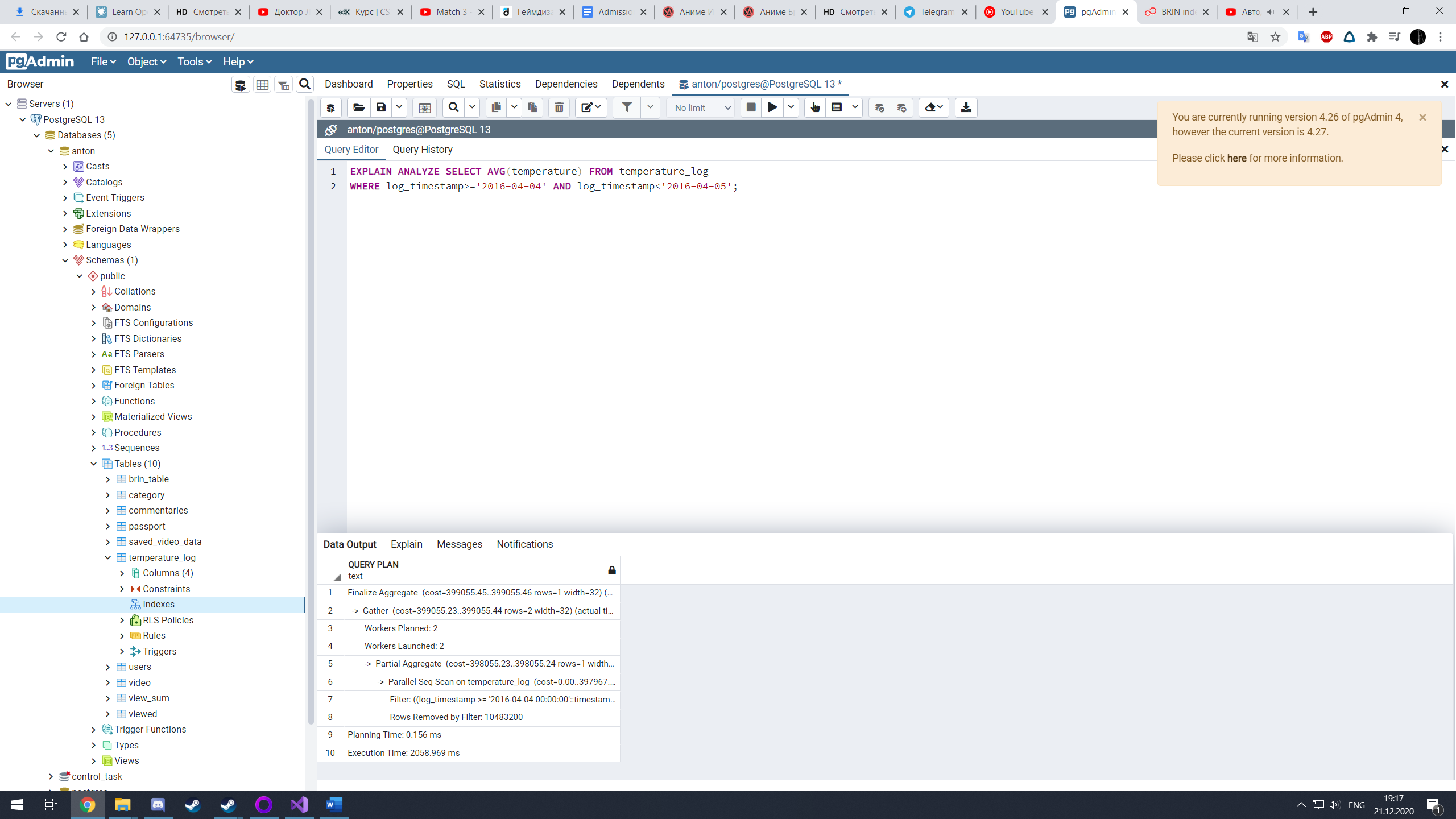
**Створення та аналіз індекса BRIN**

Iндекс BRIN використовують на данних значення яких вiдповiдає іх розташуванню в пам'яті. Була використана таблиця що має поле log\_timestamp що зберігає час отримання данних. В таблицю занесено 31536001 кортежей. Виконаний запит:

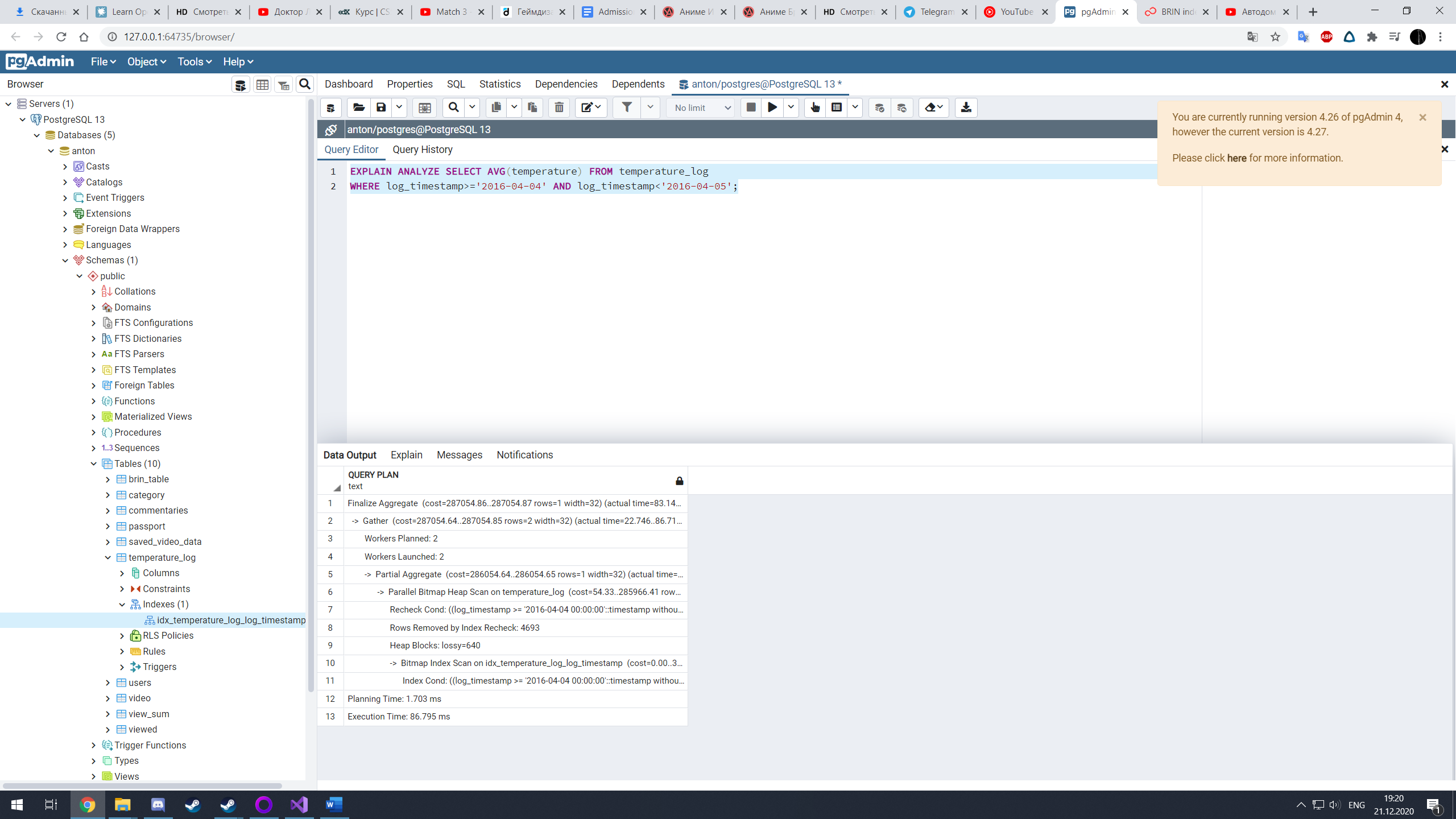
SELECT AVG(temperature) FROM temperature\_log

WHERE log\_timestamp>='2016-04-04' AND log\_timestamp<'2016-04-05';

До індексування

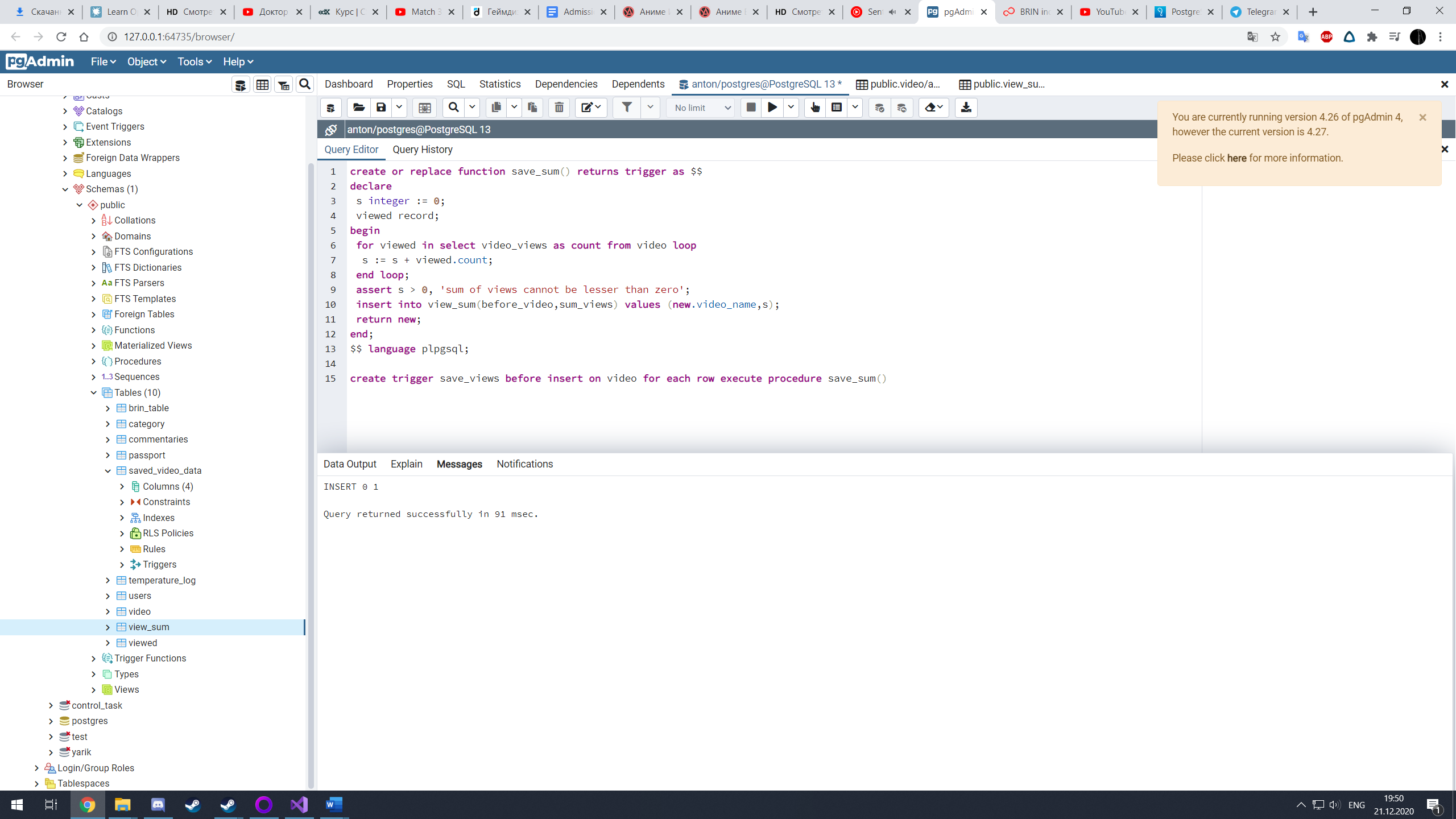


Після індексування.



**Завдання 3**

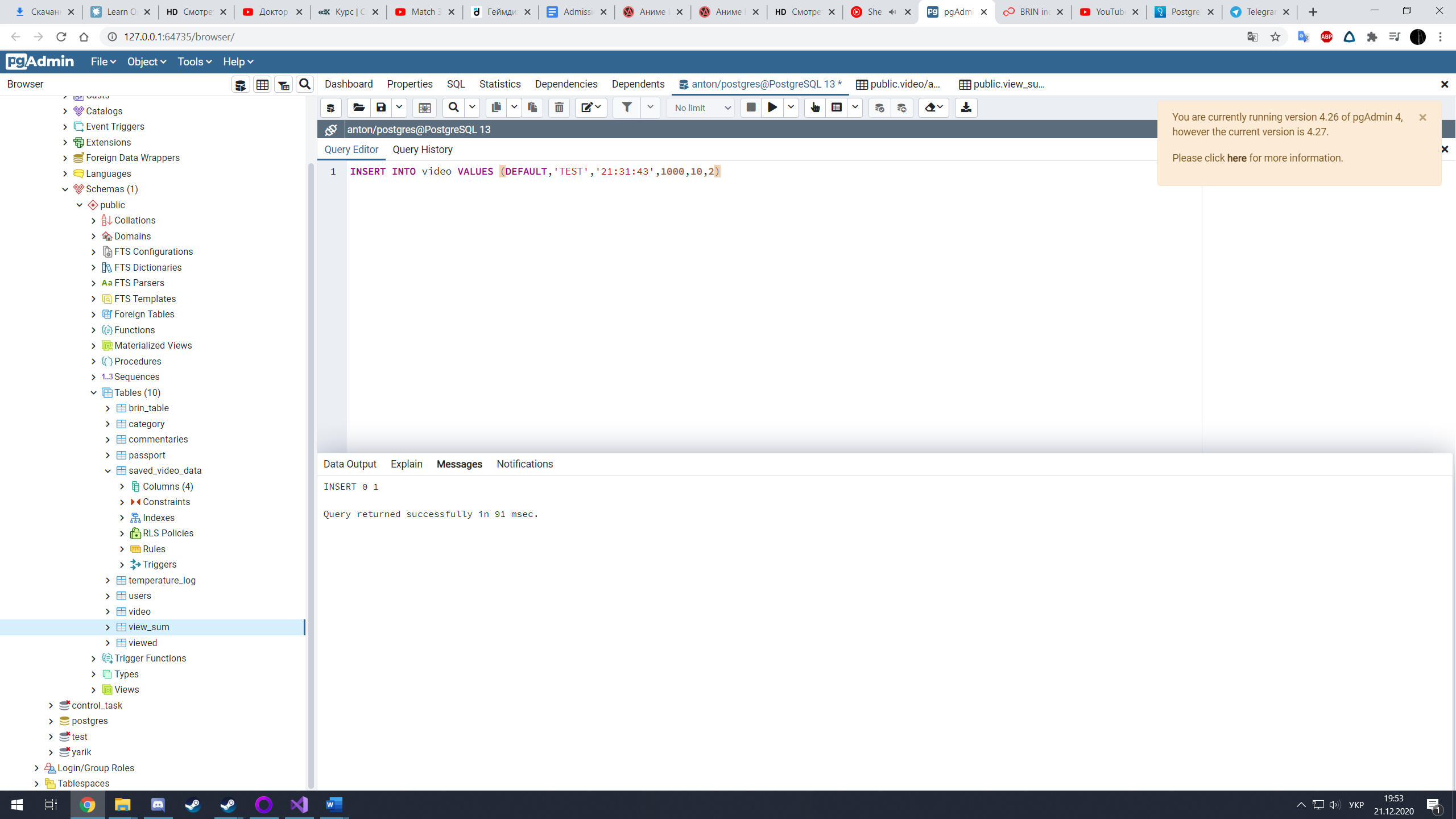
Тригер:



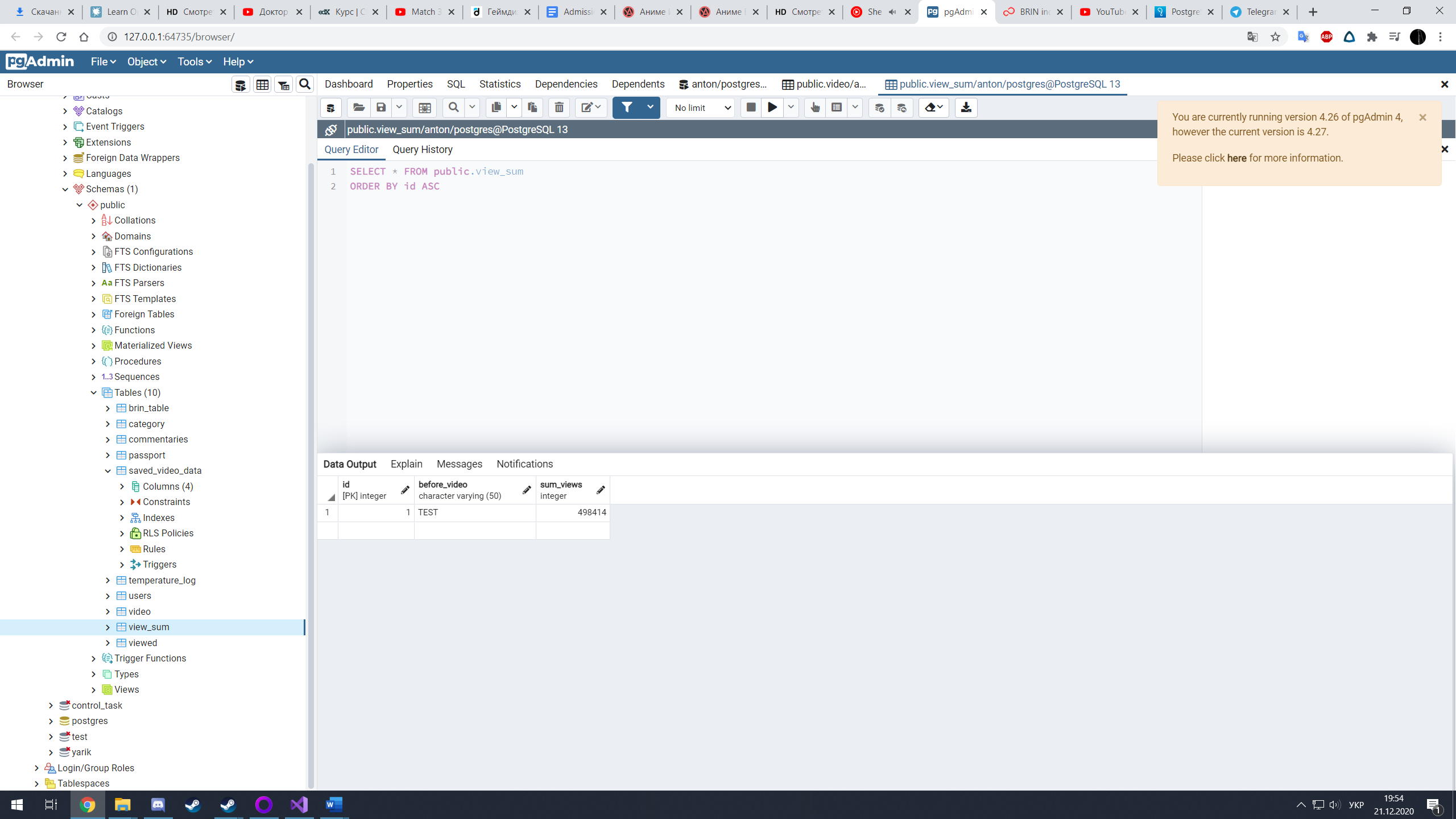
Принцип роботи:

Тригер спрацьовує перед вставкою в таблицю video та зберігає суму переглядів всіх відео до вставки та назву відео в таблицю view\_sum.

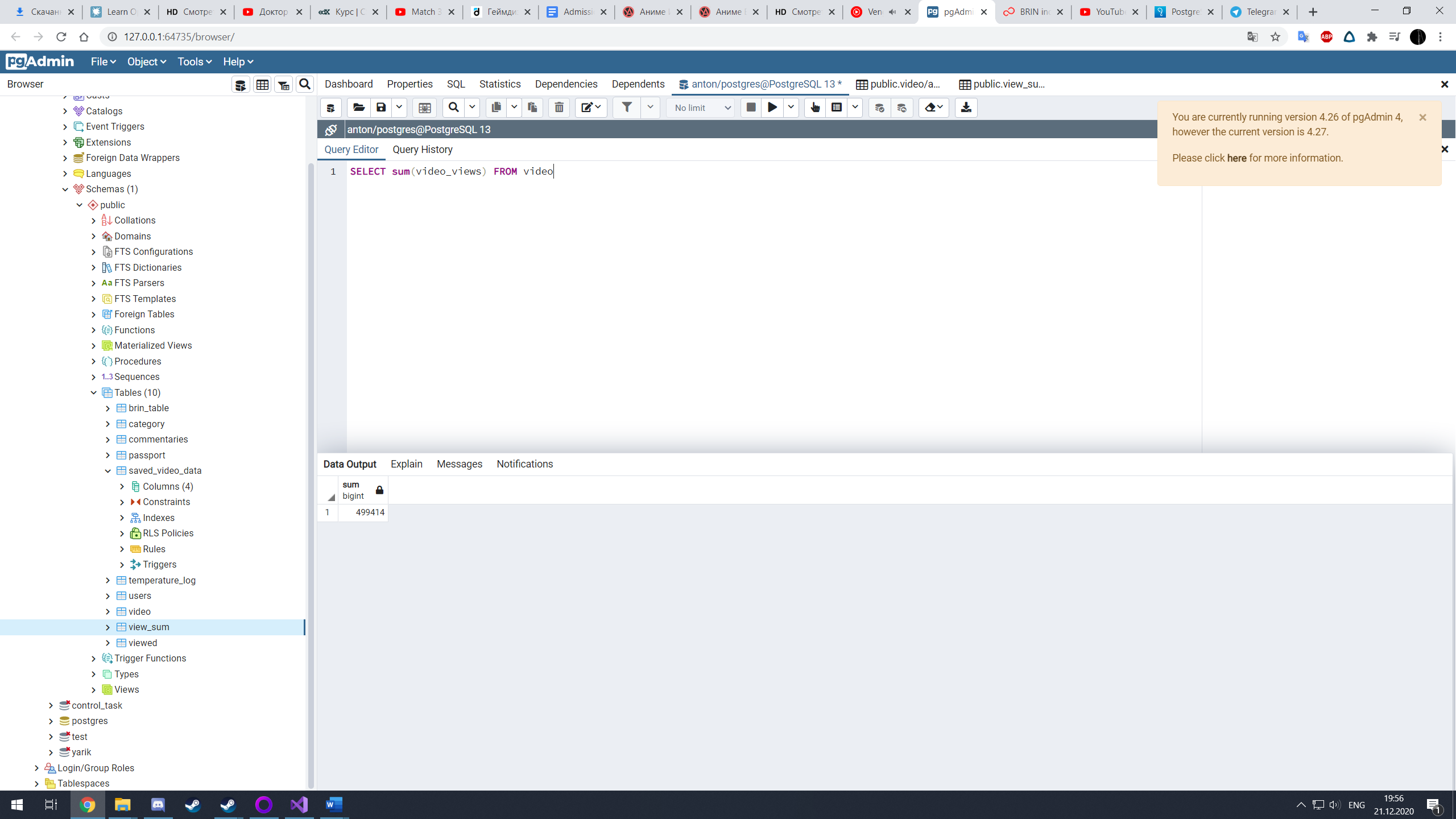
Вставляємо запис у video:



view\_sum:



Перевіримо суму:



499414 – 1000 = 498414 => Все вірно