//Calculating the ADG for each Goat
*** we don't have a max weight in a view or table yet, this needs to be derived
create or replace view maxweight as select animal_id, when_measured, max(alpha_value) as max_weight from weight group by animal_id order by animal_id;

animal_id	when_measured	max_weight

*** this is needed for future joins/calculations, so we need to make it a view and join it with the dams/kids to get both max and birthweight:
Create or replace view adg as SELECT mw.animal_id, (mw.max_weight - g.birth_weight) / DATEDIFF(m.when_measured, g.dob), g.dob AS adg FROM maxweight as mw inner join goat as g on g.animal_id = mw.animal_id;

*** this view now holds:

animal_id	adg	dob

Once we have this, we can join adg to season month on extract(month from dob) = season_month.month like:

Create or replace view adgbyseason as select a.animal_id, a.adg, sm.season from adg as a inner join season_month as sm on extract(month from a.dob) = season_month.month;

We now have:

animal_id	adg	season

And:

Create or replace view almost as Select seasonname, avg(adg) as averageADG from adgbyseason group by season;

Which will give us:

season	averageADG
1	value
2	value
3	value
4	value

Which we can join with season:
 from season as s natural join almost as a.
Select s.seasonname, a.averageADG

seasonname	averageADG
winter	value
spring	value
summer	value
fall	value

^
And that is our report _____|
Lol… seems like a lot for 4 numbers.
//Calculating the average adg:

```
SELECT
    Season, (this has to be changed to the view we are using)
    AVG((max_weight - birth_weight) / DATEDIFF(max_weight_date, dob)) AS average_adg
FROM goats_view
GROUP BY season;
```

Sorting the columns:

Calculating the ADG for each goat

```
WITH adg_calculation AS (
    SELECT
        id,
        (max_weight - birth_weight) / DATEDIFF(max_weight_date, dob) AS adg,
        season
    FROM goats_view
)
```

```
-- Sort the goats by season and calculate the average ADG for each group
SELECT
    season,
    AVG(adg) AS average_adg
FROM adg_calculation
GROUP BY season
ORDER BY season;
```

—----------

```
from django.shortcuts import render
from django.http import HttpResponse
import psycopg2
from psycopg2 import extras
# Create your views here.
def index(request):
        connection = psycopg2.connect(database="goats", user="lion", password="lion",
host="localhost", port=5432)
        curr = connection.cursor(cursor_factory = psycopg2.extras.DictCursor)
        q = 'create or replace view maxweight as select animal_id, when_measured,
max(alpha_value) as max_weight from weight group by animal_id order by animal_id;'
        curr.execute(q)
        q = 'Create or replace view adg as SELECT mw.animal_id, (mw.max_weight -
g.birth_weight) / DATEDIFF(m.when_measured, g.dob) AS adg, g.dob FROM maxweight as mw
inner join goat as g on g.animal_id = mw.animal_id;'
        curr.execute(q)
    q = 'Create or replace view adgbyseason as select a.animal_id, a.adg, sm.season from adg
as a inner join season_month as sm on extract(month from a.dob) = season_month.month;'
    curr.execute(q)
    q = 'Create or replace view almost as Select seasonname, avg(adg) as averageADG from
adgbyseason group by season;'
    curr.execute(q)
    q = 'Select s.seasonname, a.averageADG from season as s natural join almost as a;'
    curr.execute(q)



        s = curr.fetchall()
        s = [season for season in s]
        context={'s':s }
        return render(request,'seasons/index.html',context)
# Create your views here.
```