Django, window functions and paginator

Some time ago I faced a strange issue. My query was completely okay but once run by Django ORM it returned slightly modified data. Off by one value to be exact. It took me quite some time to hunt down the bug in my code so I'm writing about that so you can save your time.

What was wrong?

I had the following SQL query

```
SELECT
    "core_price"."id",
    "core_price"."item_id",
    "core_price"."datetime",
    "core_price"."price",
    (
        100.0
        * (("core_price"."price" / LEAD("core_price"."price", 1) OVER ()) -
         ↪  1)
    ) AS "change"
FROM "core_price"
WHERE "core_price"."item_id" = 193
ORDER BY "core_price"."datetime" DESC;
```

where I fetch from `core_price` table and calculate percentage change of a day gains the previous day. It's done by window `LEAD()` function. Output directly from PostgreSQL CLI looked like this:

```
+--------+---------+------------------------+--------+--------------------
 ↪  +
| id     | item_id | datetime               | price  | change              |
|--------+---------+------------------------+--------+--------------------
 ↪  |
| 849323 | 193     | 2025-01-17 00:00:00+00 | 105.57 | 5.296230316162109   |
| 849226 | 193     | 2025-01-16 00:00:00+00 | 100.26 | -0.4369378089904785 |
| 849129 | 193     | 2025-01-15 00:00:00+00 | 100.7  | 2.6189804077148438  |
| 849032 | 193     | 2025-01-14 00:00:00+00 | 98.13  | 1.4788031578063965  |
| 848957 | 193     | 2025-01-13 00:00:00+00 | 96.7   | 2.133500576019287   |
| 848956 | 193     | 2025-01-10 00:00:00+00 | 94.68  | -5.622011423110962  |
+--------+---------+------------------------+--------+--------------------
 ↪  +
```

Everything worked great until I realized that on production I was getting 500. The issue was that the first row had empty `change` value which carries the output from window function.

Django ORM query looked like this:

```
self.model.objects.annotate(
    change=100.0 * (F("price") / Window(Lag("price")) - 1)
)
.filter(item=user_item.item)
.order_by("-datetime")
```

but for some reason was giving me the following result:

```
+--------+---------+------------------------+--------+---------------------
 ↪  +
| id     | item_id | datetime               | price  | change              |
|--------+---------+------------------------+--------+---------------------
 ↪  |
| 849323 | 193     | 2025-01-17 00:00:00+00 | 105.57 | <null>              |
| 849226 | 193     | 2025-01-16 00:00:00+00 | 100.26 | -5.0298333168029785 |
| 849129 | 193     | 2025-01-15 00:00:00+00 | 100.7  | 0.4388570785522461  |
| 849032 | 193     | 2025-01-14 00:00:00+00 | 98.13  | -2.552133798599243  |
| 848957 | 193     | 2025-01-13 00:00:00+00 | 96.7   | -1.4572501182556152 |
| 848956 | 193     | 2025-01-10 00:00:00+00 | 94.68  | -2.0889341831207275 |
+--------+---------+------------------------+--------+---------------------
 ↪  +
```

How did I debug the code?

I put some breakpoints around in my class that extends `ListView`. I tried to inspect `object_list` right in my `get_context_data()` method but all I got was the first row with empty `change` column.

Well that was weird and it got even weirder once I did print the query that Django constructed and was sending to the database (`print(self.get_queryset().query)`) because the query was identical to the first query in this article above and the empty column should be populated.

What was another level of weirdness was that the other rows had strange `change` values. Like what the fuck? Then I realized that I use paginator (which I already eliminated earlier for debug purposes). Based on the level of "WTF" that I've had been into I decided to turn on SQL query logging in Django so I can see what queries are actually send to the database. Easily done by:

```
LOGGING["loggers"]["django.db.backends"] = {
    "handlers": ["console"],
    "level": "DEBUG",
```

```
    "propagate": False,
}
```

in my settings file (assumes the logging is already set). This was the key move because sometimes you think A but reality offers B.

Where is was the issue?

I was very surprised when I saw queries ending with `LIMIT 21`. Also if I ran `self.get_queryset()[0]` to debug the first problematic row I saw a query ending with `LIMIT 1`. Well here we go. The limit completely changes the window calculation. But the bug is in different part of the query - in the `ORDER BY` directive.

What's the solution?

Solution is simple - the ordering must be in `OVER()` so each window is sorted and then calculated correctly. Once a `LIMIT` is added it doesn't affect windows. So the final query is:

```
SELECT
    "core_price"."id",
    "core_price"."item_id",
    "core_price"."datetime",
    "core_price"."price",
    (
        100.0
        * (
            (
                "core_price"."price"
                / LEAD("core_price"."price", 1)
                    OVER (ORDER BY "core_price"."datetime" DESC)
            )
            - 1
        )
    ) AS "change"
FROM "core_price"
WHERE "core_price"."item_id" = 193
LIMIT 10;
```

Takeaways

The main takeaway is that once you into “WTF” moment you need to eliminate all the additional stuff that are wrapped around current issue - like the paginator. Second is to use proper debugging and third is to dig to the deepest and rawest point - in this case stop trusting ORM and seek the true SQU query that is run against the database which can be then debugged/tuned in PostgreSQL CLI.