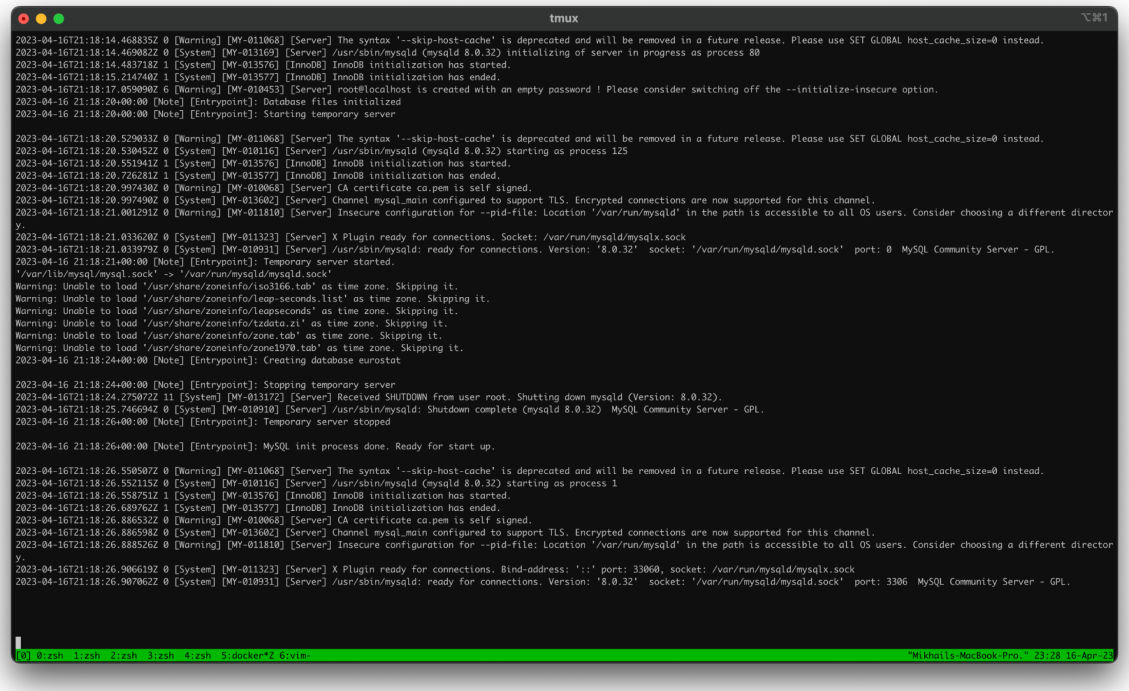
Advanced Databases Project

Assignment 3

Introduction

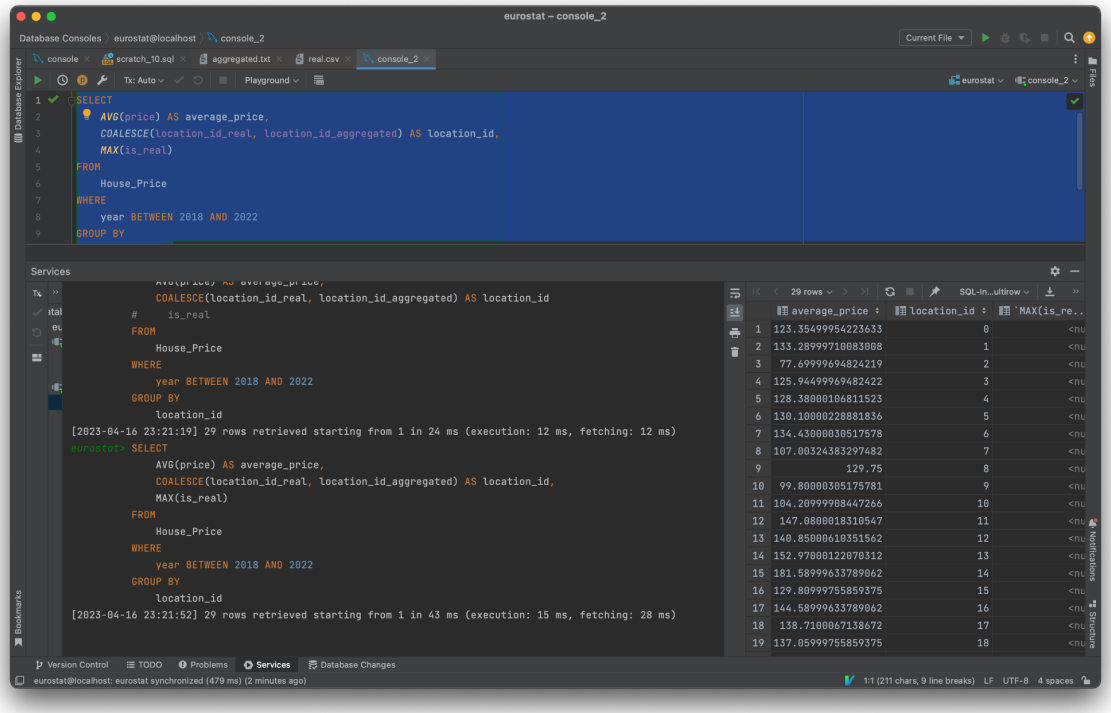
MySQL log never changed during the execution of queries even though we switched debug mode on. Consider this screenshot as an illustration of our MySQL instance running.



Working queries

1. Get average house prices for the last 5 years for every location:

```
SELECT AVG(price) AS average_price, COALESCE(location_id_real,
location_id_aggregated) AS location_id, MAX(is_real)
FROM House_Price
WHERE year BETWEEN 2018 AND 2022
GROUP BY location_id
```

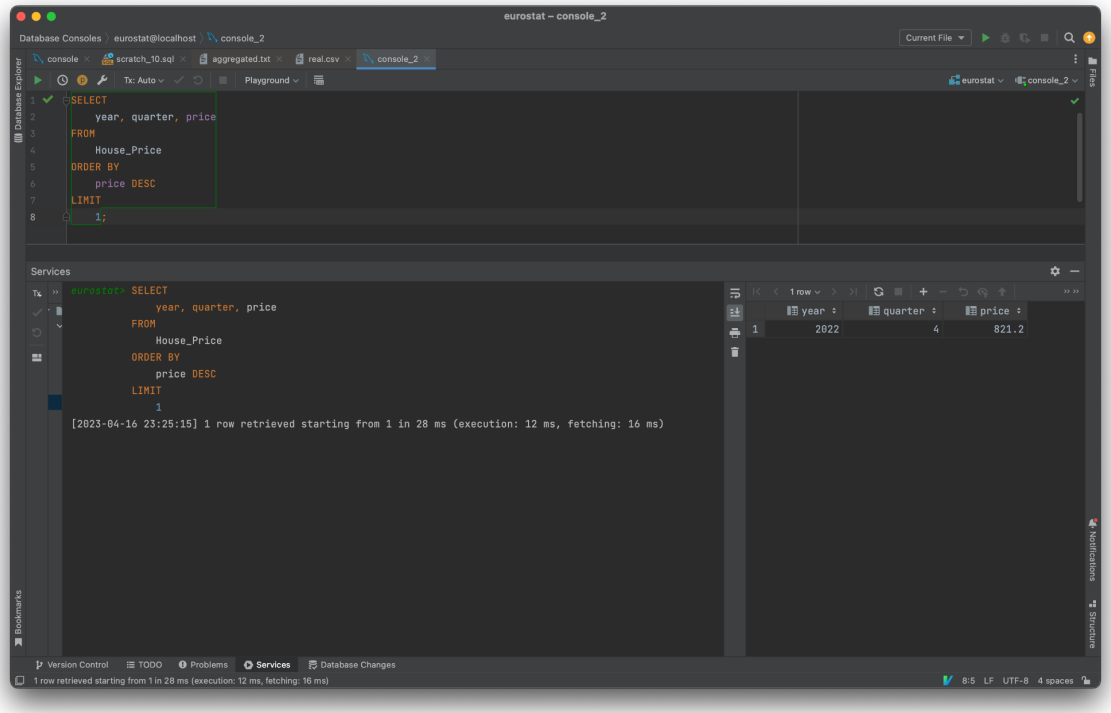


Execution log

```
-> Table scan on <temporary>  (actual time=0.252..0.255 rows=29 loops=1)
    -> Aggregate using temporary table  (actual time=0.252..0.252
rows=29 loops=1)
        -> Filter: (House_Price.`year` between 2018 and 2022)
(cost=37.25 rows=41) (actual time=0.051..0.142 rows=370 loops=1)
            -> Table scan on House_Price  (cost=37.25 rows=370) (actual
time=0.047..0.116 rows=370 loops=1)
```

2. Get the year, quarter and price when the price was maximum:

```
SELECT year, quarter, price
FROM House_Price
ORDER BY price DESC
LIMIT 1
```

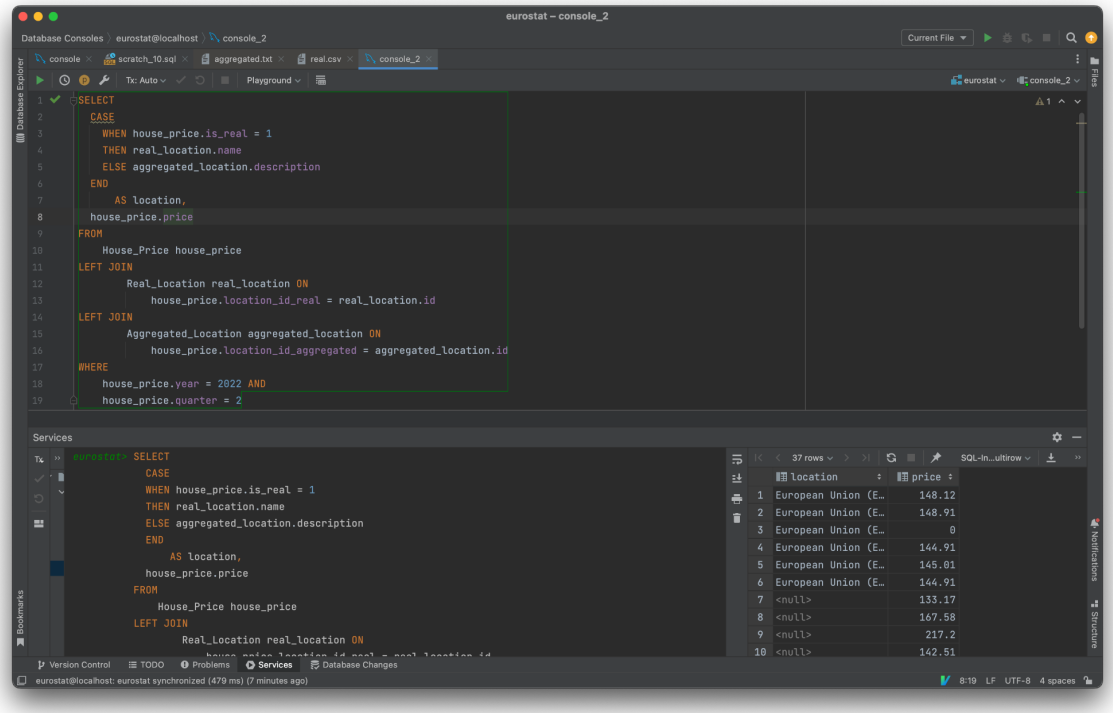


Execution log:

```
-> Limit: 1 row(s)  (cost=37.25 rows=1) (actual time=0.122..0.122 rows=1 loops=1)
    -> Sort: House_Price.price DESC, limit input to 1 row(s) per chunk (cost=37.25 rows=370) (actual time=0.121..0.121 rows=1 loops=1)
        -> Table scan on House_Price  (cost=37.25 rows=370) (actual time=0.032..0.087 rows=370 loops=1)
```

3. Get the names (descriptions) and prices for specific year and quarter:

```
SELECT
  CASE
      WHEN house_price.is_real = 1 THEN real_location.name
      ELSE aggregated_location.description
  END AS location,
  house_price.price
FROM House_Price house_price
LEFT JOIN Real_Location real_location ON house_price.location_id_real = real_location.id
LEFT JOIN Aggregated_Location aggregated_location ON house_price.location_id_aggregated = aggregated_location.id
WHERE house_price.year = 2022 AND house_price.quarter = 2
```



Execution log:

```
-> Nested loop left join  (cost=39.84 rows=4) (actual time=0.124..0.209
rows=37 loops=1)
    -> Nested loop left join  (cost=38.55 rows=4) (actual
time=0.113..0.180 rows=37 loops=1)
        -> Filter: ((house_price.`quarter` = 2) and (house_price.`year`
= 2022))  (cost=37.25 rows=4) (actual time=0.109..0.151 rows=37 loops=1)
            -> Table scan on house_price  (cost=37.25 rows=370) (actual
time=0.043..0.129 rows=370 loops=1)
        -> Single-row index lookup on real_location using PRIMARY
(id=house_price.location_id_real)  (cost=0.28 rows=1) (actual
time=0.001..0.001 rows=1 loops=37)
    -> Single-row index lookup on aggregated_location using PRIMARY
(id=house_price.location_id_aggregated)  (cost=0.28 rows=1) (actual
time=0.001..0.001 rows=0 loops=37)
```

4. Get average job vacancy ratio for the last 5 years for every location:

```
SELECT AVG(ratio) AS average_ratio, COALESCE(location_id_real,
location_id_aggregated) AS location_id, MAX(is_real) AS is_real
FROM Job_Vacancy_Ratio
WHERE year BETWEEN 2018 AND 2022
GROUP BY location_id
```

5. Get the year, quarter and ratio when the ratio was maximum:

```
SELECT year, quarter, ratio
FROM Job_Vacancy_Ratio
ORDER BY ratio DESC
LIMIT 1
```

6. Get the names (descriptions) and ratios for specific year and quarter:

```
SELECT
  CASE
    WHEN job_vacancy_ratio.is_real = 1 THEN real_location.name
    ELSE aggregated_location.description
  END AS location,
  job_vacancy_ratio.ratio
FROM Job_Vacancy_Ratio job_vacancy_ratio
LEFT JOIN Real_Location real_location ON
job_vacancy_ratio.location_id_real = real_location.id
LEFT JOIN Aggregated_Location aggregated_location ON
job_vacancy_ratio.location_id_aggregated = aggregated_location.id
WHERE job_vacancy_ratio.year = 2022 AND job_vacancy_ratio.quarter = 2
```

7. Get average consumer prices for the last 5 years for every location:

```
SELECT AVG(price) AS average_price, COALESCE(location_id_real,
location_id_aggregated) AS location_id, MAX(is_real) AS is_real
FROM Consumer_Price
WHERE year BETWEEN 2018 AND 2022
GROUP BY location_id
```

8. Get the year, quarter and price when the price was maximum:

```
SELECT year, quarter, price
FROM Consumer_Price
ORDER BY price DESC
LIMIT 1
```

9. Get the names (descriptions) and prices for specific year and quarter:

```
SELECT
  CASE
      WHEN consumer_price.is_real = 1 THEN real_location.name
      ELSE aggregated_location.description
  END AS location,
  house_price.price
```

```sql
FROM Consumer_Price consumer_price
LEFT JOIN Real_Location real_location ON consumer_price.location_id_real
= real_location.id
LEFT JOIN Aggregated_Location aggregated_location ON
consumer_price.location_id_aggregated = aggregated_location.id
WHERE consumer_price.year = 2022 AND consumer_price.quarter = 2
```

Conclusion

In the application, some values (year or quarter) will be parametrized.

Since our previous assignment submission we have discovered rows containing duplicated year and quarter attributes. Therefore we have added an “id” column to “House_Price” and “Job_Vacancy_Ratio” tables and made iit the primary key.