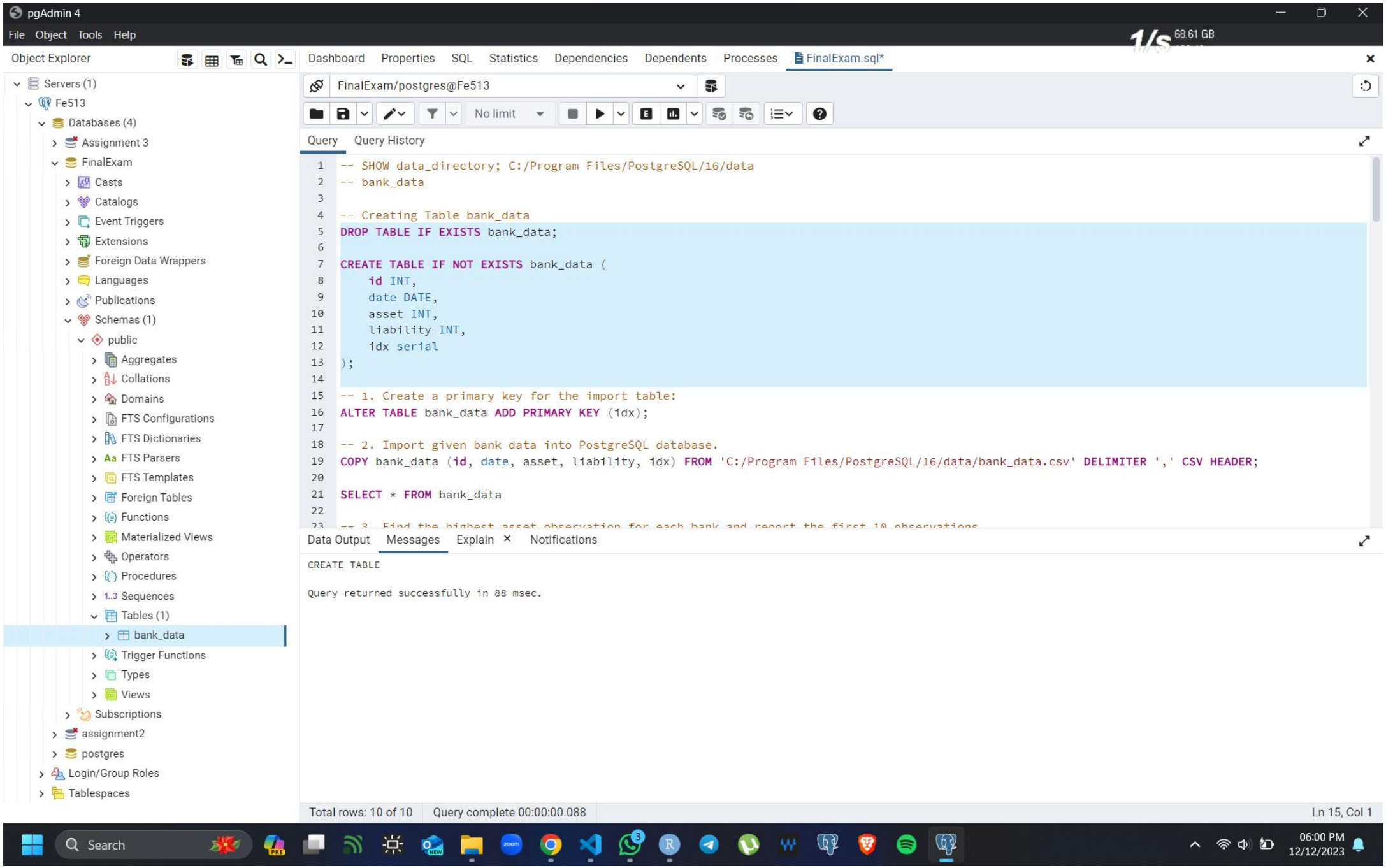
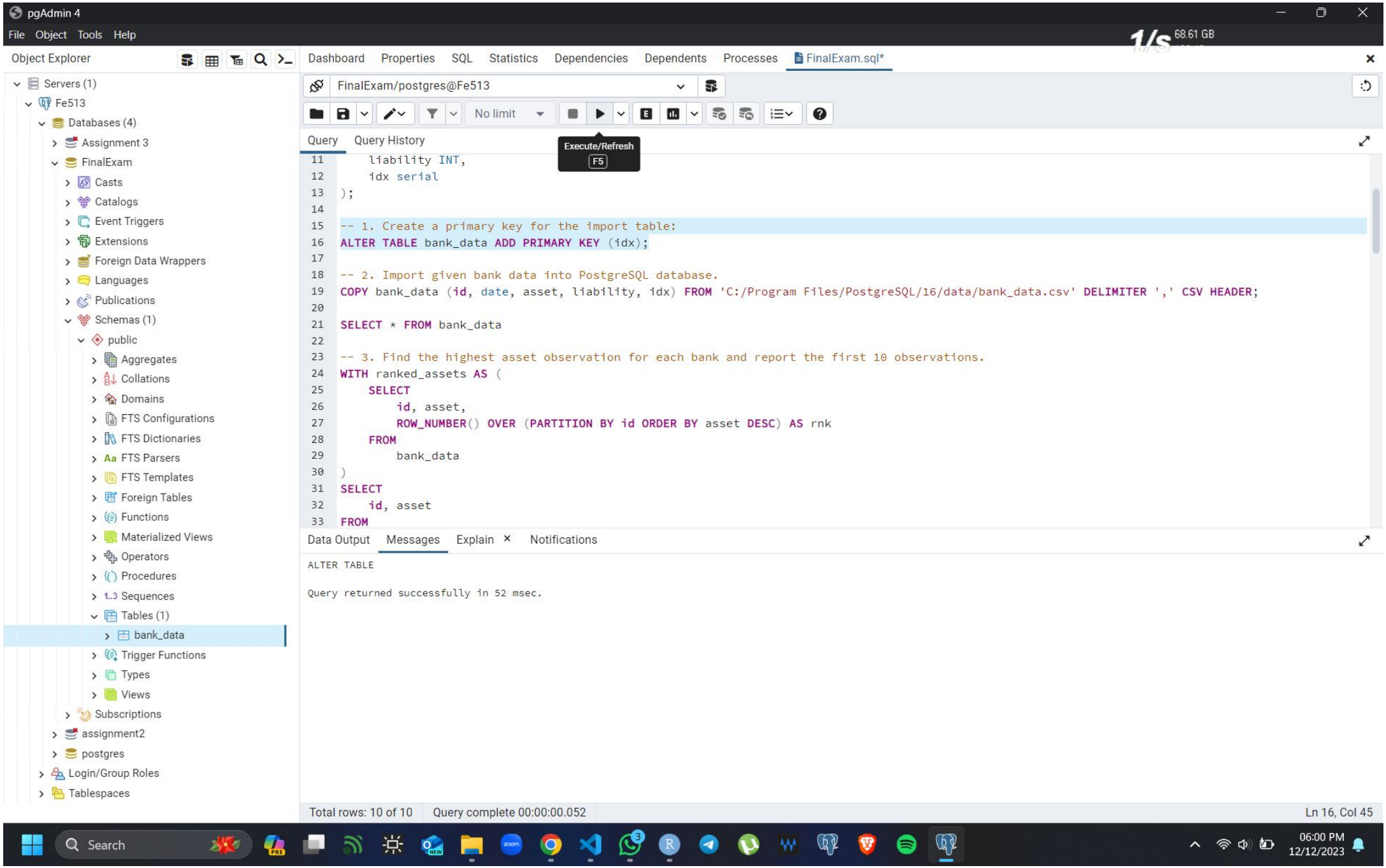

pgAdmin 4
File Object Tools Help
68.61 GB
Object Explorer
Dashboard Properties SQL Statistics Dependencies Dependents Processes FinalExam.sql*
Servers (1)
Fe513
Databases (4)
Assignment 3
FinalExam
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (1)
bank_data
Trigger Functions
Types
Views
Subscriptions
assignment2
postgres
Login/Group Roles
Tablespaces
FinalExam/postgres@Fe513
No limit
Query Query History
1 -- SHOW data_directory; C:/Program Files/PostgreSQL/16/data
2 -- bank_data
3
4 -- Creating Table bank_data
5 DROP TABLE IF EXISTS bank_data;
6
7 CREATE TABLE IF NOT EXISTS bank_data (
8 id INT,
9 date DATE,
10 asset INT,
11 liability INT,
12 idx serial
13);
14
15 -- 1. Create a primary key for the import table:
16 ALTER TABLE bank_data ADD PRIMARY KEY (idx);
17
18 -- 2. Import given bank data into PostgreSQL database.
19 COPY bank_data (id, date, asset, liability, idx) FROM 'C:/Program Files/PostgreSQL/16/data/bank_data.csv' DELIMITER ',' CSV HEADER;
20
21 SELECT * FROM bank_data
22
Data Output Messages Explain Notifications
CREATE TABLE
Query returned successfully in 88 msec.
Total rows: 10 of 10
Query complete 00:00:00.088
Ln 15, Col 1
Search
06:00 PM
12/12/2023


pgAdmin 4
File Object Tools Help
68.61 GB
Object Explorer
Dashboard Properties SQL Statistics Dependencies Dependents Processes FinalExam.sql*
Servers (1)
Fe513
Databases (4)
Assignment 3
FinalExam
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (1)
bank_data
Trigger Functions
Types
Views
Subscriptions
assignment2
postgres
Login/Group Roles
Tablespaces
FinalExam/postgres@Fe513
No limit
Execute/Refresh
F5
Query Query History
liability INT,
idx serial
);
-- 1. Create a primary key for the import table:
ALTER TABLE bank_data ADD PRIMARY KEY (idx);
-- 2. Import given bank data into PostgreSQL database.
COPY bank_data (id, date, asset, liability, idx) FROM 'C:/Program Files/PostgreSQL/16/data/bank_data.csv' DELIMITER ',' CSV HEADER;
SELECT * FROM bank_data
-- 3. Find the highest asset observation for each bank and report the first 10 observations.
WITH ranked_assets AS (
SELECT
id, asset,
ROW_NUMBER() OVER (PARTITION BY id ORDER BY asset DESC) AS rnk
FROM
bank_data
)
SELECT
id, asset
FROM
Data Output Messages Explain Notifications
ALTER TABLE
Query returned successfully in 52 msec.
Total rows: 10 of 10
Query complete 00:00:00.052
Ln 16, Col 45
Search
06:00 PM
12/12/2023

pgAdmin 4

File Object Tools Help

Object Explorer

- Servers (1)
  - Fe513
    - Databases (4)
      - Assignment 3
      - FinalExam
        - Casts
        - Catalogs
        - Event Triggers
        - Extensions
        - Foreign Data Wrappers
        - Languages
        - Publications
        - Schemas (1)
          - public
            - Aggregates
            - Collations
            - Domains
            - FTS Configurations
            - FTS Dictionaries
            - FTS Parsers
            - FTS Templates
            - Foreign Tables
            - Functions
            - Materialized Views
            - Operators
            - Procedures
            - Sequences
            - Tables (1)
              - bank_data
            - Trigger Functions
            - Types
            - Views
        - Subscriptions
      - assignment2
      - postgres
    - Login/Group Roles
    - Tablespaces

Dashboard Properties SQL Statistics Dependencies Dependents Processes FinalExam.sql*

FinalExam/postgres@Fe513

No limit

Execute/Refresh F5

Query Query History

```
11      liability INT,
12      idx serial
13  );
14
15  -- 1. Create a primary key for the import table:
16  ALTER TABLE bank_data ADD PRIMARY KEY (idx);
17
18  -- 2. Import given bank data into PostgreSQL database.
19  COPY bank_data (id, date, asset, liability, idx) FROM 'C:/Program Files/PostgreSQL/16/data/bank_data.csv' DELIMITER ',' CSV HEADER;
20
21  SELECT * FROM bank_data
22
23  -- 3. Find the highest asset observation for each bank and report the first 10 observations.
24  WITH ranked_assets AS (
25      SELECT
26          id, asset,
27          ROW_NUMBER() OVER (PARTITION BY id ORDER BY asset DESC) AS rnk
28      FROM
29          bank_data
30  )
31  SELECT
32      id, asset
33  FROM
```

Data Output Messages Explain Notifications

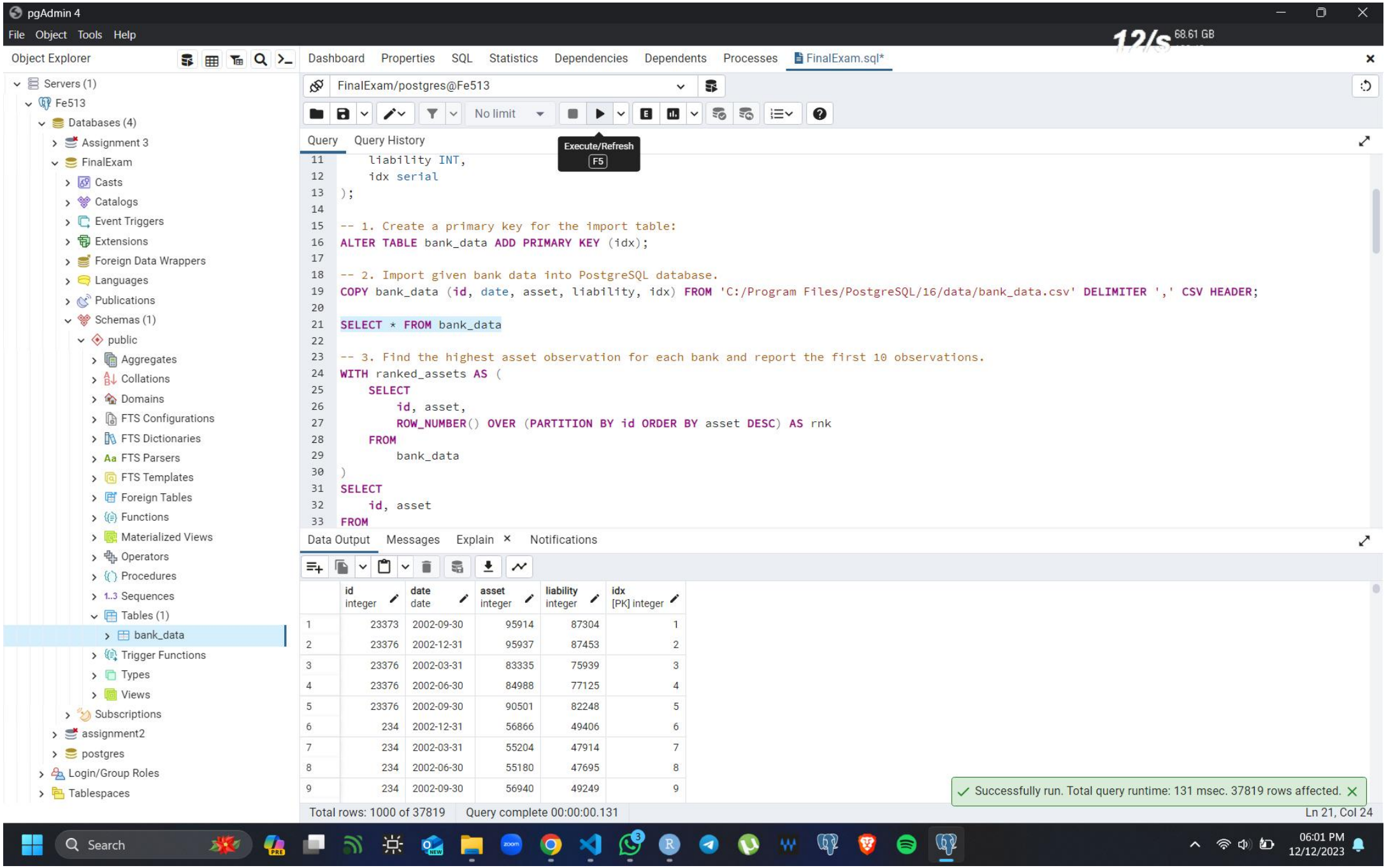
```
COPY 37819

Query returned successfully in 235 msec.
```

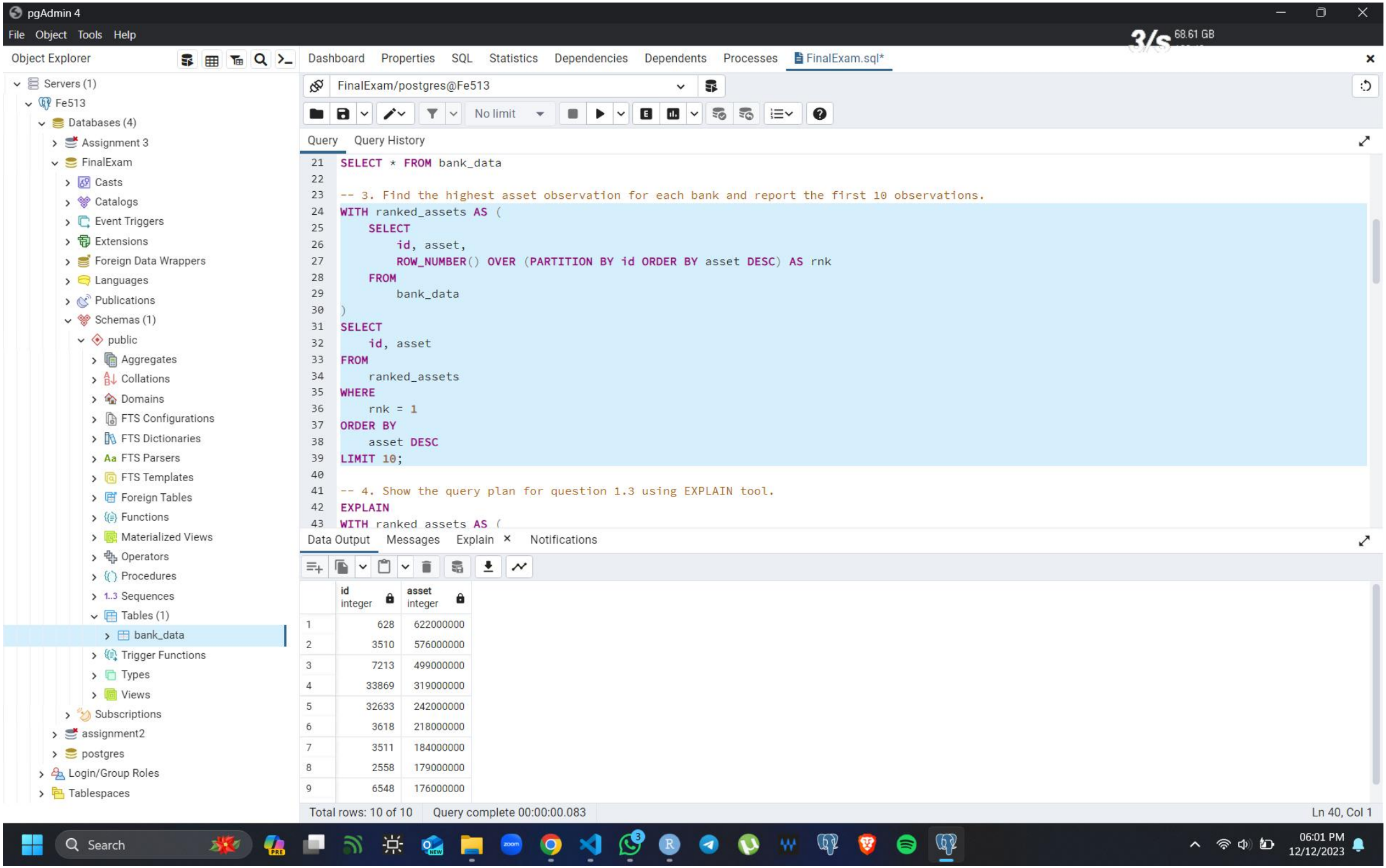
Query returned successfully in 235 msec.

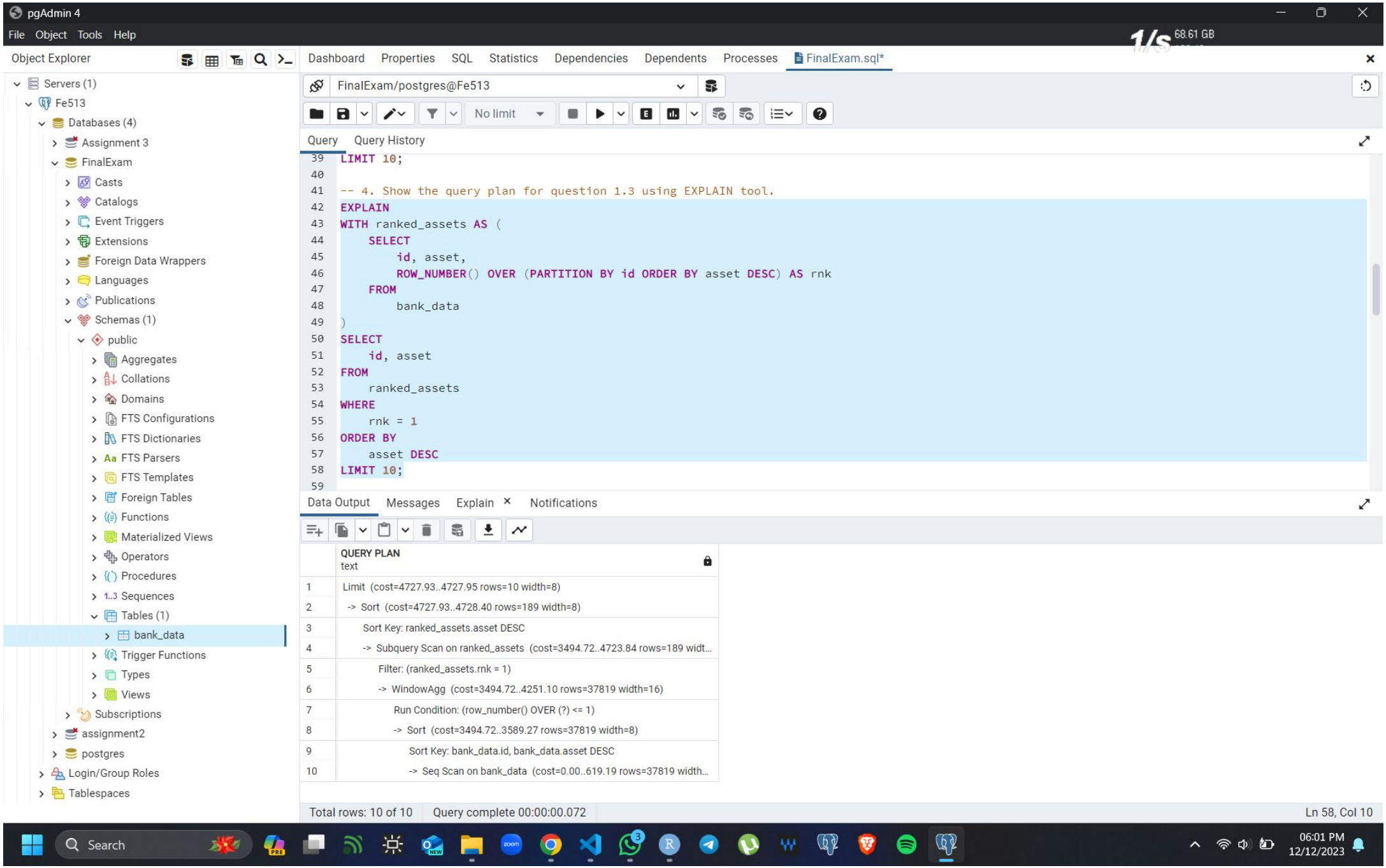
Total rows: 10 of 10 Query complete 00:00:00.235 Ln 20, Col 1

Search

06:01 PM 12/12/2023


pgAdmin 4
File Object Tools Help
Object Explorer
Servers (1)
Fe513
Databases (4)
Assignment 3
FinalExam
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (1)
bank_data
Trigger Functions
Types
Views
Subscriptions
assignment2
postgres
Login/Group Roles
Tablespaces
Dashboard Properties SQL Statistics Dependencies Dependents Processes FinalExam.sql*
FinalExam/postgres@Fe513
No limit
Execute/Refresh
F5
Query Query History
liability INT,
idx serial
);
-- 1. Create a primary key for the import table:
ALTER TABLE bank_data ADD PRIMARY KEY (idx);
-- 2. Import given bank data into PostgreSQL database.
COPY bank_data (id, date, asset, liability, idx) FROM 'C:/Program Files/PostgreSQL/16/data/bank_data.csv' DELIMITER ',' CSV HEADER;
SELECT * FROM bank_data
-- 3. Find the highest asset observation for each bank and report the first 10 observations.
WITH ranked_assets AS (
SELECT
id, asset,
ROW_NUMBER() OVER (PARTITION BY id ORDER BY asset DESC) AS rnk
FROM
bank_data
)
SELECT
id, asset
FROM
Data Output Messages Explain Notifications
id integer
date date
asset integer
liability integer
idx [PK] integer
1 23373 2002-09-30 95914 87304 1
2 23376 2002-12-31 95937 87453 2
3 23376 2002-03-31 83335 75939 3
4 23376 2002-06-30 84988 77125 4
5 23376 2002-09-30 90501 82248 5
6 234 2002-12-31 56866 49406 6
7 234 2002-03-31 55204 47914 7
8 234 2002-06-30 55180 47695 8
9 234 2002-09-30 56940 49249 9
Successfully run. Total query runtime: 131 msec. 37819 rows affected.
Total rows: 1000 of 37819
Query complete 00:00:00.131
Ln 21, Col 24
Search
06:01 PM
12/12/2023


pgAdmin 4
File Object Tools Help
Object Explorer
Servers (1)
Fe513
Databases (4)
Assignment 3
FinalExam
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (1)
bank_data
Trigger Functions
Types
Views
Subscriptions
assignment2
postgres
Login/Group Roles
Tablespaces
Dashboard Properties SQL Statistics Dependencies Dependents Processes FinalExam.sql*
FinalExam/postgres@Fe513
No limit
Query Query History
21 SELECT * FROM bank_data
22
23 -- 3. Find the highest asset observation for each bank and report the first 10 observations.
24 WITH ranked_assets AS (
25 SELECT
26 id, asset,
27 ROW_NUMBER() OVER (PARTITION BY id ORDER BY asset DESC) AS rnk
28 FROM
29 bank_data
30)
31 SELECT
32 id, asset
33 FROM
34 ranked_assets
35 WHERE
36 rnk = 1
37 ORDER BY
38 asset DESC
39 LIMIT 10;
40
41 -- 4. Show the query plan for question 1.3 using EXPLAIN tool.
42 EXPLAIN
43 WITH ranked assets AS (
Data Output Messages Explain Notifications
id integer asset integer
1 628 622000000
2 3510 576000000
3 7213 499000000
4 33869 319000000
5 32633 242000000
6 3618 218000000
7 3511 184000000
8 2558 179000000
9 6548 176000000
Total rows: 10 of 10 Query complete 00:00:00.083 Ln 40, Col 1
Search
06:01 PM
12/12/2023


pgAdmin 4
File Object Tools Help
68.61 GB
Object Explorer
Servers (1)
Fe513
Databases (4)
Assignment 3
FinalExam
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (1)
bank_data
Trigger Functions
Types
Views
Subscriptions
assignment2
postgres
Login/Group Roles
Tablespaces
Dashboard Properties SQL Statistics Dependencies Dependents Processes FinalExam.sql*
FinalExam/postgres@Fe513
No limit
Query Query History
39 LIMIT 10;
40
41 -- 4. Show the query plan for question 1.3 using EXPLAIN tool.
42 EXPLAIN
43 WITH ranked_assets AS (
44 SELECT
45 id, asset,
46 ROW_NUMBER() OVER (PARTITION BY id ORDER BY asset DESC) AS rnk
47 FROM
48 bank_data
49)
50 SELECT
51 id, asset
52 FROM
53 ranked_assets
54 WHERE
55 rnk = 1
56 ORDER BY
57 asset DESC
58 LIMIT 10;
59
Data Output Messages Explain Notifications
QUERY PLAN
text
1 Limit (cost=4727.93..4727.95 rows=10 width=8)
2 -> Sort (cost=4727.93..4728.40 rows=189 width=8)
3 Sort Key: ranked_assets.asset DESC
4 -> Subquery Scan on ranked_assets (cost=3494.72..4723.84 rows=189 widt...
5 Filter: (ranked_assets.rnk = 1)
6 -> WindowAgg (cost=3494.72..4251.10 rows=37819 width=16)
7 Run Condition: (row_number() OVER (?) <= 1)
8 -> Sort (cost=3494.72..3589.27 rows=37819 width=8)
9 Sort Key: bank_data.id, bank_data.asset DESC
10 -> Seq Scan on bank_data (cost=0.00..619.19 rows=37819 width...
Total rows: 10 of 10
Query complete 00:00:00.072
Ln 58, Col 10
Search
06:01 PM
12/12/2023

pgAdmin 4

File Object Tools Help

Object Explorer

- Servers (1)
  - Fe513
    - Databases (4)
      - Assignment 3
      - FinalExam
        - Casts
        - Catalogs
        - Event Triggers
        - Extensions
        - Foreign Data Wrappers
        - Languages
        - Publications
        - Schemas (1)
          - public
            - Aggregates
            - Collations
            - Domains
            - FTS Configurations
            - FTS Dictionaries
            - FTS Parsers
            - FTS Templates
            - Foreign Tables
            - Functions
            - Materialized Views
            - Operators
            - Procedures
            - Sequences
            - Tables (1)
              - bank_data
            - Trigger Functions
            - Types
            - Views
        - Subscriptions
      - assignment2
      - postgres
    - Login/Group Roles
    - Tablespaces

Dashboard Properties SQL Statistics Dependencies Dependents Processes FinalExam.sql*

FinalExam/postgres@Fe513

No limit

Query Query History

```
39  LIMIT 10;
40
41  -- 4. Show the query plan for question 1.3 using EXPLAIN tool.
42  EXPLAIN
43  WITH ranked_assets AS (
44      SELECT
45          id, asset,
46          ROW_NUMBER() OVER (PARTITION BY id ORDER BY asset DESC) AS rnk
47      FROM
48          bank_data
49  )
50  SELECT
51      id, asset
52  FROM
53      ranked_assets
54  WHERE
55      rnk = 1
56  ORDER BY
57      asset DESC
58  LIMIT 10;
59
```

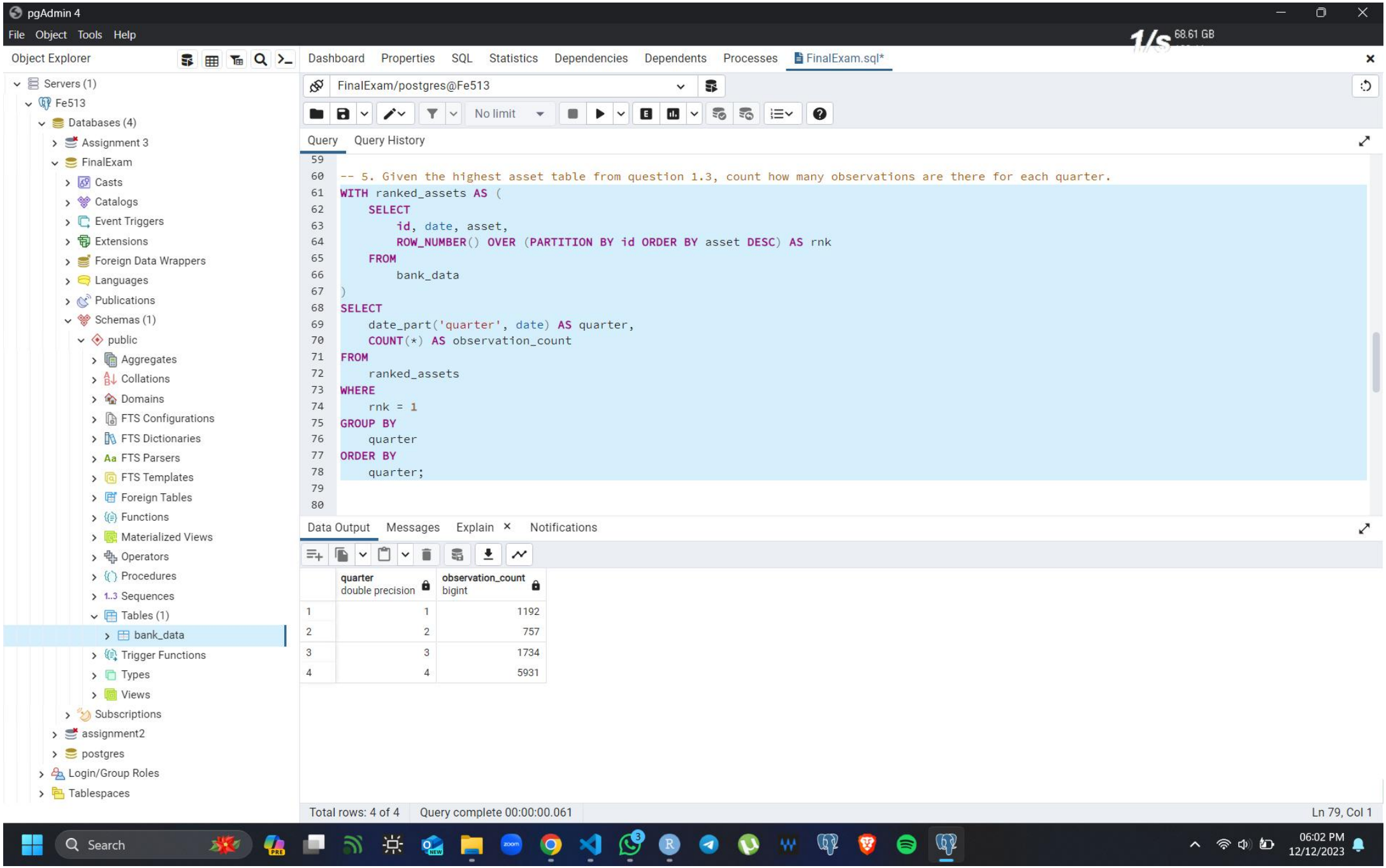
Data Output Messages Explain Notifications

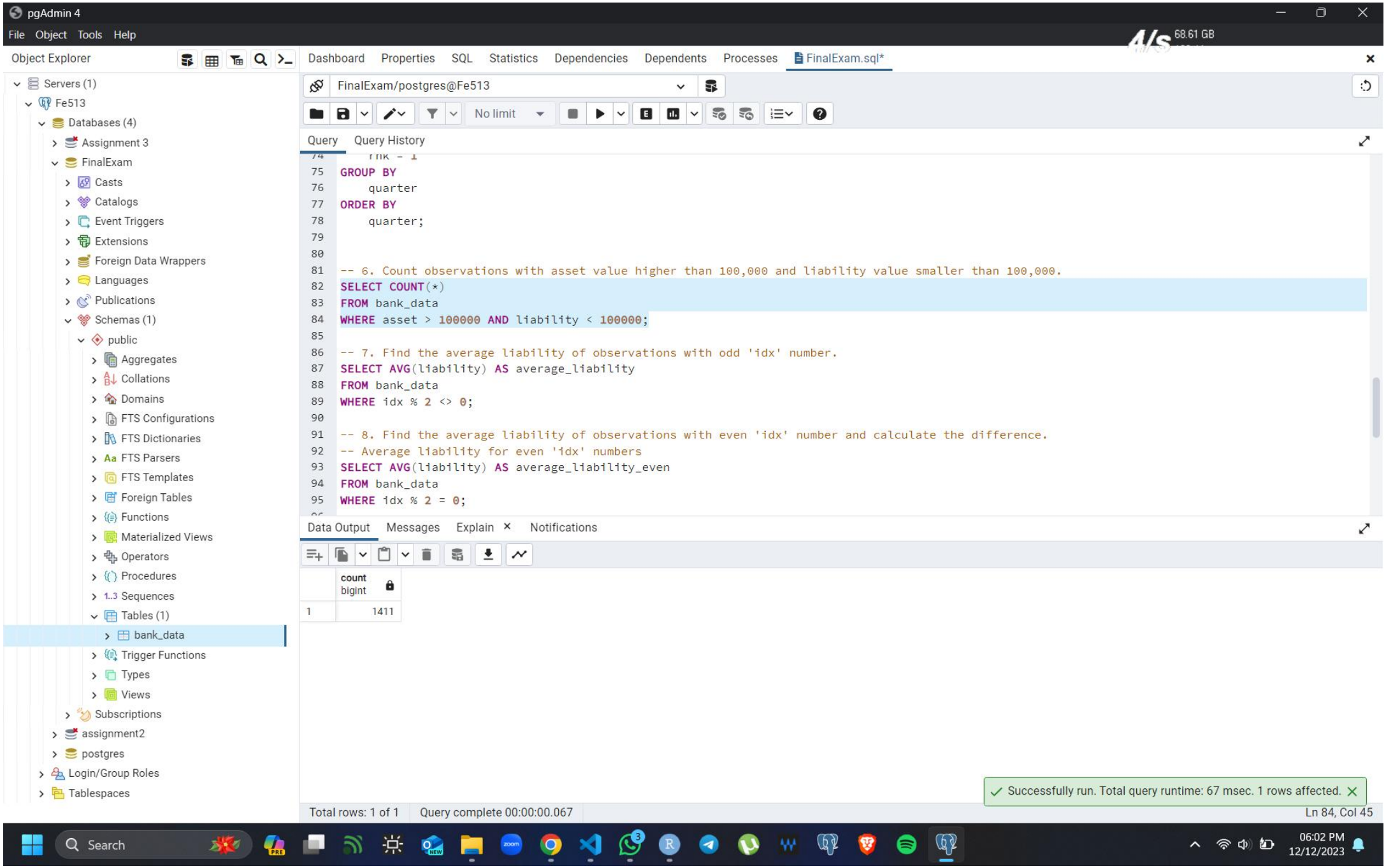
Graphical Analysis Statistics

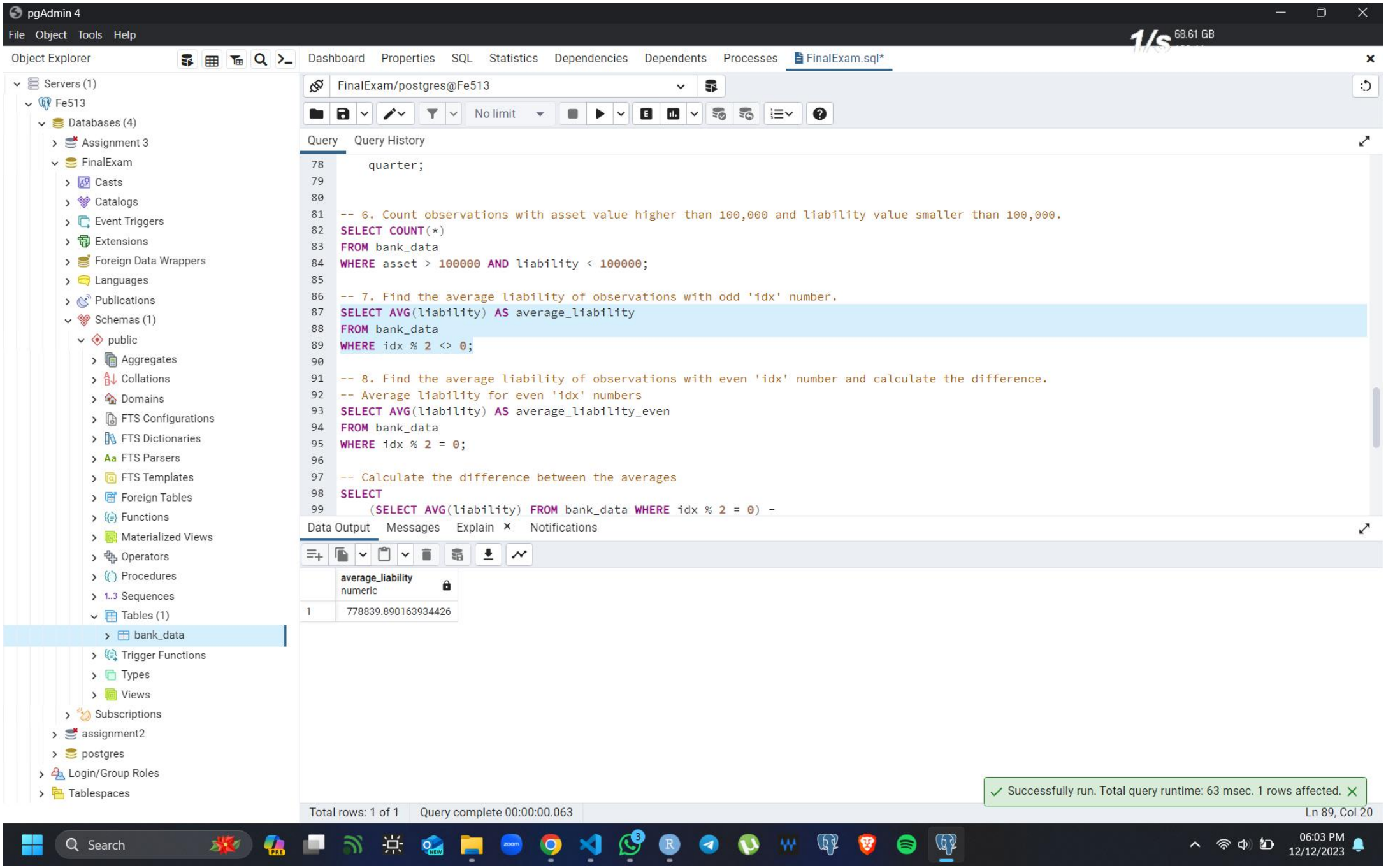
Total rows: 1 of 1 Query complete 00:00:00.067 Ln 58, Col 10

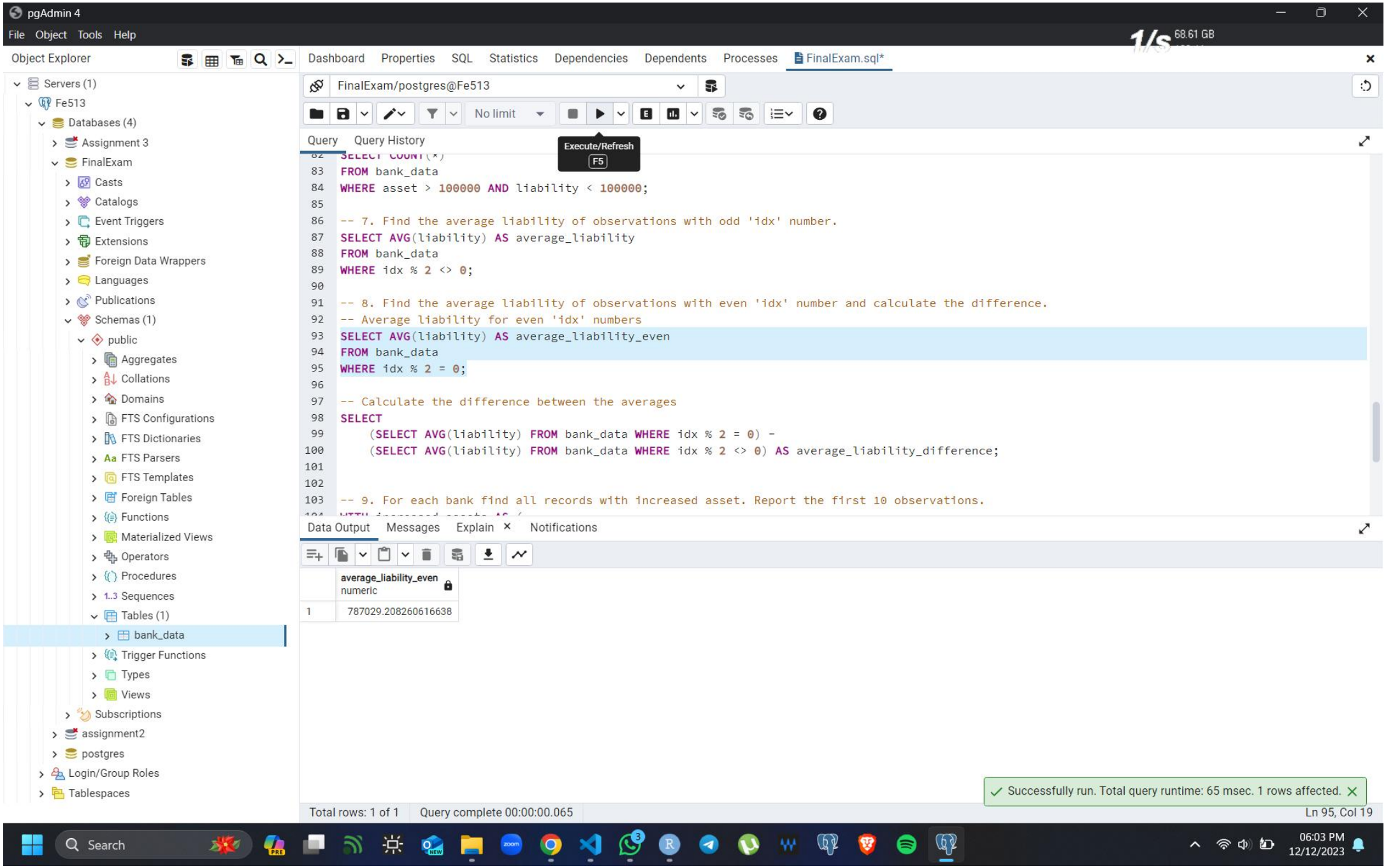
Search

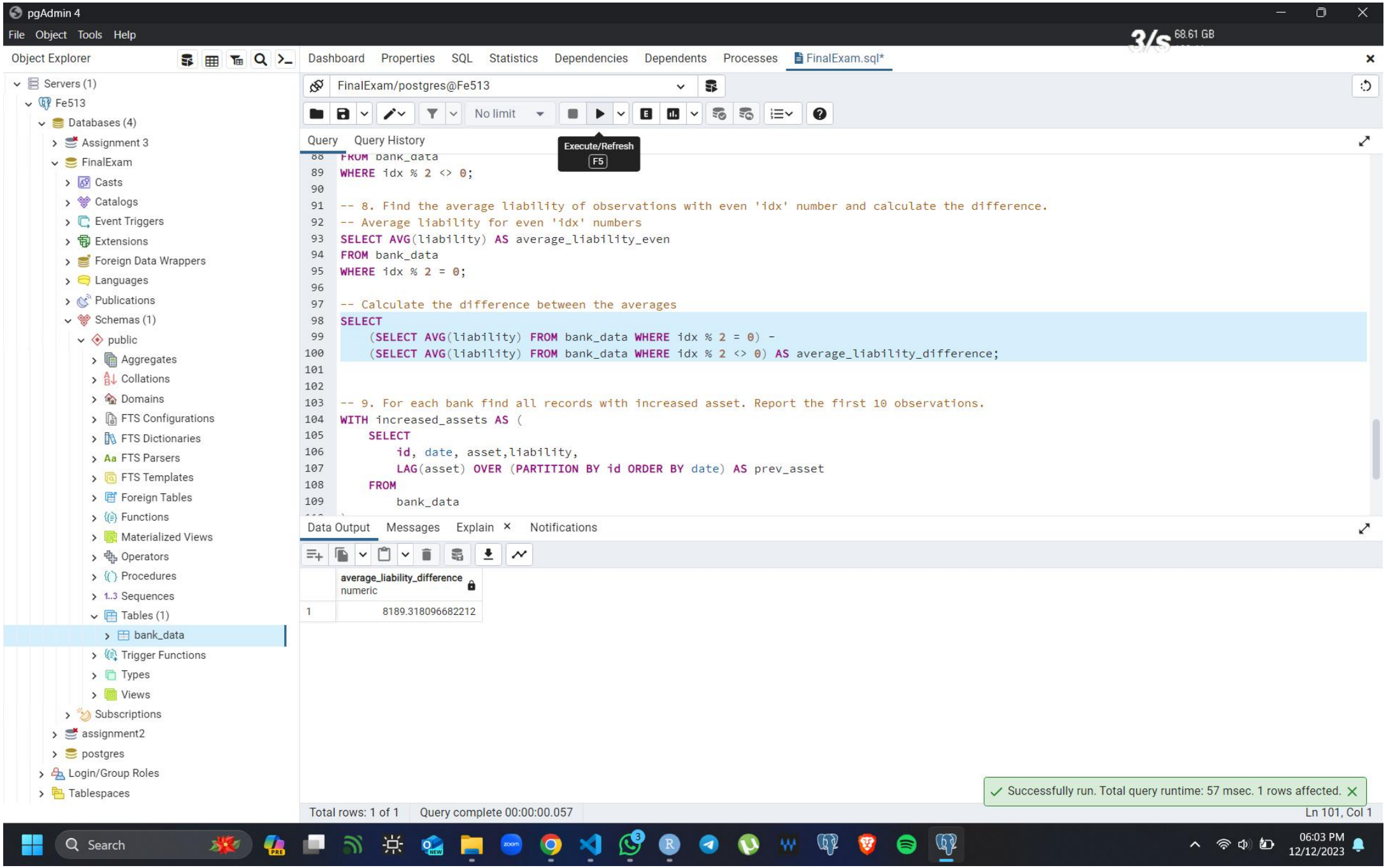
06:02 PM
12/12/2023

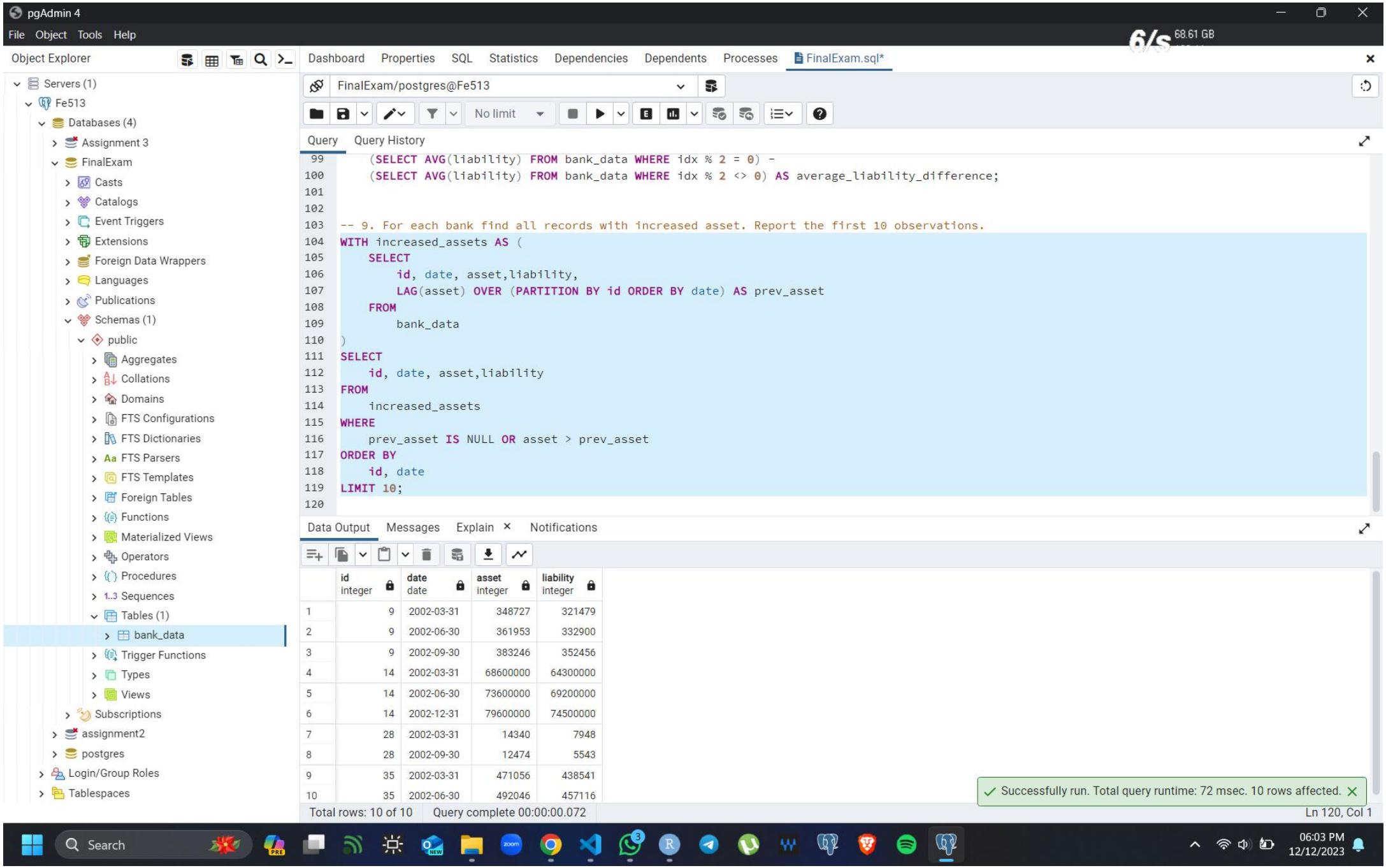

pgAdmin 4
File Object Tools Help
68.61 GB
Object Explorer
Dashboard Properties SQL Statistics Dependencies Dependents Processes FinalExam.sql*
Servers (1)
Fe513
Databases (4)
Assignment 3
FinalExam
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (1)
bank_data
Trigger Functions
Types
Views
Subscriptions
assignment2
postgres
Login/Group Roles
Tablespaces
FinalExam/postgres@Fe513
No limit
Query Query History
59
60 -- 5. Given the highest asset table from question 1.3, count how many observations are there for each quarter.
61 WITH ranked_assets AS (
62 SELECT
63 id, date, asset,
64 ROW_NUMBER() OVER (PARTITION BY id ORDER BY asset DESC) AS rnk
65 FROM
66 bank_data
67)
68 SELECT
69 date_part('quarter', date) AS quarter,
70 COUNT(*) AS observation_count
71 FROM
72 ranked_assets
73 WHERE
74 rnk = 1
75 GROUP BY
76 quarter
77 ORDER BY
78 quarter;
79
80
Data Output Messages Explain Notifications
quarter double precision
observation_count bigint
1 1 1192
2 2 757
3 3 1734
4 4 5931
Total rows: 4 of 4
Query complete 00:00:00.061
Ln 79, Col 1
Search
06:02 PM
12/12/2023


pgAdmin 4
File Object Tools Help
68.61 GB
Object Explorer
Dashboard Properties SQL Statistics Dependencies Dependents Processes FinalExam.sql*
Servers (1)
Fe513
Databases (4)
Assignment 3
FinalExam
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (1)
bank_data
Trigger Functions
Types
Views
Subscriptions
assignment2
postgres
Login/Group Roles
Tablespaces
FinalExam/postgres@Fe513
No limit
Query Query History
rnk = 1
GROUP BY
quarter
ORDER BY
quarter;
-- 6. Count observations with asset value higher than 100,000 and liability value smaller than 100,000.
SELECT COUNT(*)
FROM bank_data
WHERE asset > 100000 AND liability < 100000;
-- 7. Find the average liability of observations with odd 'idx' number.
SELECT AVG(liability) AS average_liability
FROM bank_data
WHERE idx % 2 <> 0;
-- 8. Find the average liability of observations with even 'idx' number and calculate the difference.
-- Average liability for even 'idx' numbers
SELECT AVG(liability) AS average_liability_even
FROM bank_data
WHERE idx % 2 = 0;
Data Output Messages Explain Notifications
count
bigint
1
1411
Successfully run. Total query runtime: 67 msec. 1 rows affected.
Total rows: 1 of 1
Query complete 00:00:00.067
Ln 84, Col 45
Search
06:02 PM
12/12/2023


pgAdmin 4
File Object Tools Help
68.61 GB
Object Explorer
Servers (1)
Fe513
Databases (4)
Assignment 3
FinalExam
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (1)
bank_data
Trigger Functions
Types
Views
Subscriptions
assignment2
postgres
Login/Group Roles
Tablespaces
Dashboard Properties SQL Statistics Dependencies Dependents Processes FinalExam.sql*
FinalExam/postgres@Fe513
No limit
Query Query History
78 quarter;
79
80
81 -- 6. Count observations with asset value higher than 100,000 and liability value smaller than 100,000.
82 SELECT COUNT(*)
83 FROM bank_data
84 WHERE asset > 100000 AND liability < 100000;
85
86 -- 7. Find the average liability of observations with odd 'idx' number.
87 SELECT AVG(liability) AS average_liability
88 FROM bank_data
89 WHERE idx % 2 <> 0;
90
91 -- 8. Find the average liability of observations with even 'idx' number and calculate the difference.
92 -- Average liability for even 'idx' numbers
93 SELECT AVG(liability) AS average_liability_even
94 FROM bank_data
95 WHERE idx % 2 = 0;
96
97 -- Calculate the difference between the averages
98 SELECT
99 (SELECT AVG(liability) FROM bank_data WHERE idx % 2 = 0) -
Data Output Messages Explain Notifications
average_liability
numeric
1 778839.890163934426
Successfully run. Total query runtime: 63 msec. 1 rows affected.
Total rows: 1 of 1
Query complete 00:00:00.063
Ln 89, Col 20
Search
06:03 PM
12/12/2023


pgAdmin 4
File Object Tools Help
68.61 GB
Object Explorer
Servers (1)
Fe513
Databases (4)
Assignment 3
FinalExam
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (1)
bank_data
Trigger Functions
Types
Views
Subscriptions
assignment2
postgres
Login/Group Roles
Tablespaces
Dashboard Properties SQL Statistics Dependencies Dependents Processes FinalExam.sql*
FinalExam/postgres@Fe513
No limit
Execute/Refresh
F5
Query Query History
SELECT COUNT(*)
FROM bank_data
WHERE asset > 100000 AND liability < 100000;
-- 7. Find the average liability of observations with odd 'idx' number.
SELECT AVG(liability) AS average_liability
FROM bank_data
WHERE idx % 2 <> 0;
-- 8. Find the average liability of observations with even 'idx' number and calculate the difference.
-- Average liability for even 'idx' numbers
SELECT AVG(liability) AS average_liability_even
FROM bank_data
WHERE idx % 2 = 0;
-- Calculate the difference between the averages
SELECT
(SELECT AVG(liability) FROM bank_data WHERE idx % 2 = 0) -
(SELECT AVG(liability) FROM bank_data WHERE idx % 2 <> 0) AS average_liability_difference;
-- 9. For each bank find all records with increased asset. Report the first 10 observations.
Data Output Messages Explain Notifications
average_liability_even
numeric
1
787029.208260616638
Successfully run. Total query runtime: 65 msec. 1 rows affected.
Total rows: 1 of 1
Query complete 00:00:00.065
Ln 95, Col 19
Search
06:03 PM
12/12/2023


pgAdmin 4
File Object Tools Help
Object Explorer
Servers (1)
Fe513
Databases (4)
Assignment 3
FinalExam
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (1)
bank_data
Trigger Functions
Types
Views
Subscriptions
assignment2
postgres
Login/Group Roles
Tablespaces
Dashboard Properties SQL Statistics Dependencies Dependents Processes FinalExam.sql*
FinalExam/postgres@Fe513
No limit
Execute/Refresh
F5
Query Query History
FROM bank_data
WHERE idx % 2 <> 0;
-- 8. Find the average liability of observations with even 'idx' number and calculate the difference.
-- Average liability for even 'idx' numbers
SELECT AVG(liability) AS average_liability_even
FROM bank_data
WHERE idx % 2 = 0;
-- Calculate the difference between the averages
SELECT
(SELECT AVG(liability) FROM bank_data WHERE idx % 2 = 0) -
(SELECT AVG(liability) FROM bank_data WHERE idx % 2 <> 0) AS average_liability_difference;
-- 9. For each bank find all records with increased asset. Report the first 10 observations.
WITH increased_assets AS (
SELECT
id, date, asset,liability,
LAG(asset) OVER (PARTITION BY id ORDER BY date) AS prev_asset
FROM
bank_data
Data Output Messages Explain Notifications
average_liability_difference
numeric
1
8189.318096682212
Successfully run. Total query runtime: 57 msec. 1 rows affected.
Total rows: 1 of 1
Query complete 00:00:00.057
Ln 101, Col 1
Search
06:03 PM
12/12/2023


pgAdmin 4
File Object Tools Help
Object Explorer
Servers (1)
Fe513
Databases (4)
Assignment 3
FinalExam
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (1)
bank_data
Trigger Functions
Types
Views
Subscriptions
assignment2
postgres
Login/Group Roles
Tablespaces
Dashboard Properties SQL Statistics Dependencies Dependents Processes FinalExam.sql*
FinalExam/postgres@Fe513
No limit
Query Query History
(SELECT AVG(liability) FROM bank_data WHERE idx % 2 = 0) -
(SELECT AVG(liability) FROM bank_data WHERE idx % 2 <> 0) AS average_liability_difference;
-- 9. For each bank find all records with increased asset. Report the first 10 observations.
WITH increased_assets AS (
SELECT
id, date, asset,liability,
LAG(asset) OVER (PARTITION BY id ORDER BY date) AS prev_asset
FROM
bank_data
)
SELECT
id, date, asset,liability
FROM
increased_assets
WHERE
prev_asset IS NULL OR asset > prev_asset
ORDER BY
id, date
LIMIT 10;
Data Output Messages Explain Notifications
id integer
date date
asset integer
liability integer
1 9 2002-03-31 348727 321479
2 9 2002-06-30 361953 332900
3 9 2002-09-30 383246 352456
4 14 2002-03-31 68600000 64300000
5 14 2002-06-30 73600000 69200000
6 14 2002-12-31 79600000 74500000
7 28 2002-03-31 14340 7948
8 28 2002-09-30 12474 5543
9 35 2002-03-31 471056 438541
10 35 2002-06-30 492046 457116
Successfully run. Total query runtime: 72 msec. 10 rows affected.
Total rows: 10 of 10
Query complete 00:00:00.072
Ln 120, Col 1
Search
06:03 PM
12/12/2023

Final Exam FE513

Shubham Narkhede

2023-12-12

```
if(!require("quantmod")){ # check for package existence
  install.packages("quantmod")
}
```

```
## Loading required package: quantmod
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: TTR
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library("quantmod")
```

```
if(!require("zoo")){ # check for package existence
  install.packages("zoo")
}
library("zoo")
```

```
if(!require("ggplot2")){ # check for package existence
  install.packages("ggplot2")
}
```

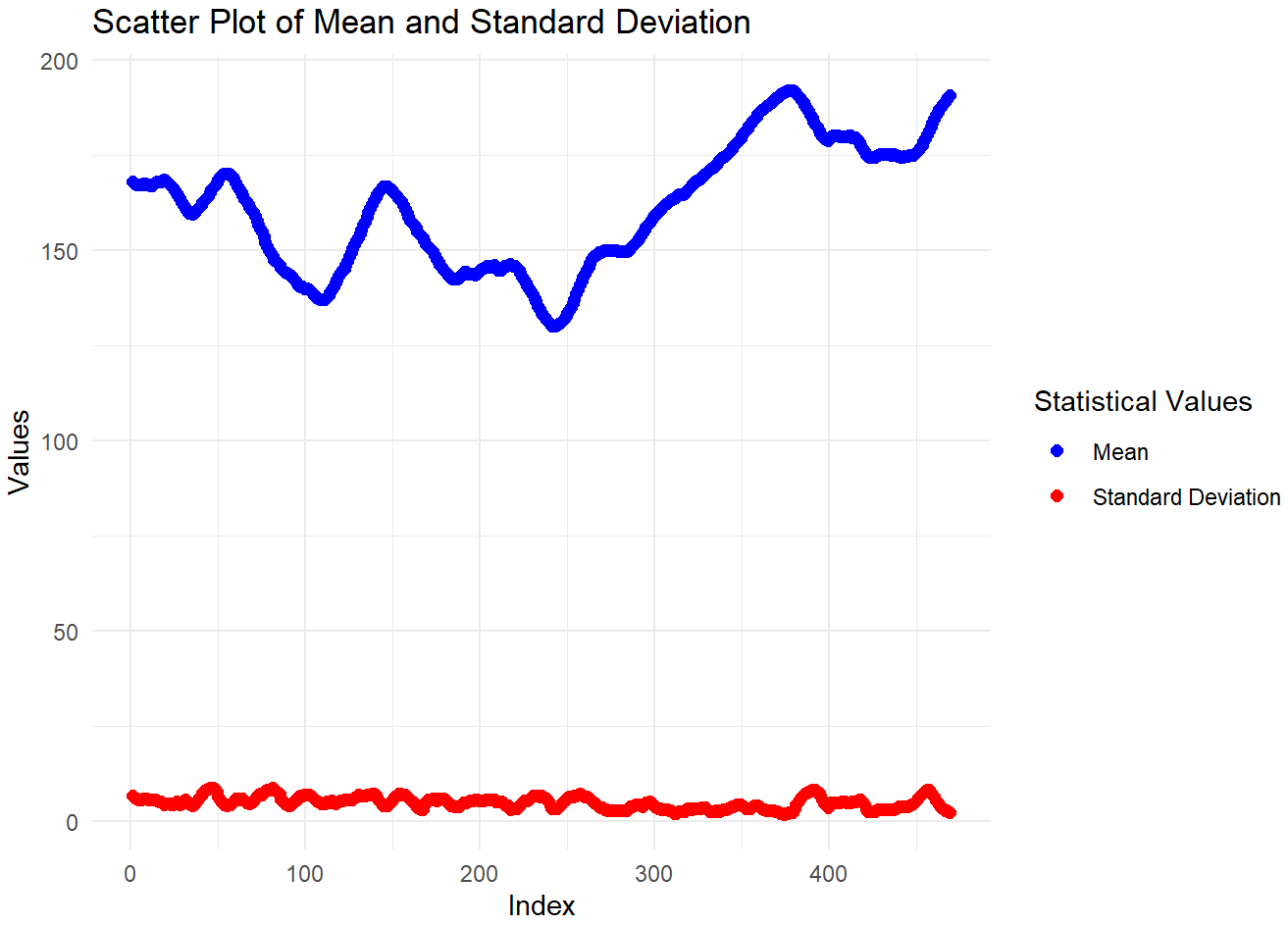
```
## Loading required package: ggplot2
```

```
library(ggplot2)
```

Question 2

```
get_stock_statistics <- function(ticker, start_date, end_date, rolling_window) {
  # Step 1: Download daily stock data
  start_date = as.Date(start_date)
  end_date = as.Date(end_date)
  getSymbols(ticker, src = "yahoo", from = start_date, to = end_date)
  stocks = get(ticker)

  # Step 2: Get the adjusted close price
  stocks = stocks[, 6]

  # Step 3: Perform rolling window estimation
  count = length(stocks)
  avg = vector()
  standard_dev = vector()

  for (i in 1:(count - rolling_window + 1)) {
    arr = stocks[i:(i + rolling_window - 1)]
    avg = c(avg, mean(arr))
    standard_dev = c(standard_dev, sd(arr))
  }

  # Step 4: Store the statistical result into a dataframe
  df = data.frame(Index = seq_along(avg), Mean = avg, Standard_Deviation = standard_dev)

  # Step 5: Plot the statistical dataframe using a scatter plot
  gg <- ggplot(df, aes(x = Index)) +
    geom_point(aes(y = Mean, color = "Mean"), size = 2) +
    geom_point(aes(y = Standard_Deviation, color = "Standard Deviation"), size = 2) +
    labs(title = "Scatter Plot of Mean and Standard Deviation",
         x = "Index",
         y = "Values") +
    scale_color_manual(name = "Statistical Values",
                       values = c("Mean" = "blue", "Standard Deviation" = "red")) +
    theme_minimal()

  # Explicitly print the plot
  print(gg)

  # Step 6: Return the statistical dataframe
  return(df)
}
```

```
# Step 7: Test the function with suitable parameters
start_date <- "2022-01-01"
end_date <- Sys.Date()
stock_ticker <- "AAPL"
window_size <- 20

result_df <- get_stock_statistics(stock_ticker, start_date, end_date, window_size)
```




```
head(result_df,10)
```

```
##    Index     Mean Standard_Deviation
## 1      1 167.9426           6.684206
## 2      2 167.5768           6.172897
## 3      3 167.3860           5.897305
## 4      4 167.2861           5.814565
## 5      5 167.3163           5.831221
## 6      6 167.3019           5.824011
## 7      7 167.4434           5.932663
## 8      8 167.5138           6.011213
## 9      9 167.3561           5.884624
## 10    10 167.1912           5.845491
```

Question 3

```
if(!require("RPostgreSQL")){ # check for package existence
  install.packages("RPostgreSQL")
}
```

```
## Loading required package: RPostgreSQL
```

```
## Loading required package: DBI
```

```
library("RPostgreSQL")
```

```
library(RPostgreSQL)
library(DBI)

# 1. Make a connection to your local PostgreSQL database.
con <- dbConnect(RPostgres::Postgres(),
                 dbname = "FinalExam",
                 host = "localhost",
                 port = 5432,
                 user = "postgres",
                 password = '123')

# 2. Query the PostgreSQL database via API to get the original bank data.
query <- "SELECT * FROM bank_data"
bank_data_df <- dbGetQuery(con, query)
head(bank_data_df)
```

```
##      id       date asset liability idx
## 1 23373 2002-09-30 95914     87304   1
## 2 23376 2002-12-31 95937     87453   2
## 3 23376 2002-03-31 83335     75939   3
## 4 23376 2002-06-30 84988     77125   4
## 5 23376 2002-09-30 90501     82248   5
## 6   234 2002-12-31 56866     49406   6
```

```
if(!require("dplyr")){ # check for package existence
  install.packages("dplyr")
}
```

```
## Loading required package: dplyr
```

```
##
## ########################## Warning from 'xts' package ##########################
## #                                                                              #
## # The dplyr lag() function breaks how base R's lag() function is supposed to  #
## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or       #
## # source() into this session won't work correctly.                            #
## #                                                                              #
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop           #
## # dplyr from breaking base R's lag() function.                                #
## #                                                                              #
## # Code in packages is not affected. It's protected by R's namespace mechanism #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
## #                                                                              #
## ################################################################################
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:xts':
##
##     first, last
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library("dplyr")
```

```
# 3. Calculate asset growth rate for each quarter and each bank.
bank_data_df$date <- as.Date(bank_data_df$date)

bank_data_df <- bank_data_df[order(bank_data_df$id, bank_data_df$date), ]  # Ensure data is sort
ed by bank id and date


# Calculate asset growth rate excluding the first quarter
bank_data_df_growth_rate <- bank_data_df %>%
  arrange(id, date) %>%
  group_by(id) %>%
  mutate(
    previous_asset = lag(asset),
    asset_growth_rate = (asset - previous_asset) / previous_asset
  ) %>%
  filter(!is.na(asset_growth_rate))

# Display the first 10 rows of the resulting data frame
head(bank_data_df_growth_rate, 10)
```

```
## # A tibble: 10 × 7
## # Groups:   id [4]
##       id date          asset liability   idx previous_asset asset_growth_rate
##    <int> <date>        <int>     <int> <int>          <int>             <dbl>
##  1     9 2002-06-30   361953    332900 20913         348727            0.0379
##  2     9 2002-09-30   383246    352456 20914         361953            0.0588
##  3     9 2002-12-31   371812    340365 20911         383246           -0.0298
##  4    14 2002-06-30 73600000  69200000 27335       68600000            0.0729
##  5    14 2002-09-30 72800000  68200000 27336       73600000           -0.0109
##  6    14 2002-12-31 79600000  74500000 27333       72800000            0.0934
##  7    28 2002-06-30    12049      5354  3938          14340           -0.160 
##  8    28 2002-09-30    12474      5543  3939          12049            0.0353
##  9    35 2002-06-30   492046    457116 12624         471056            0.0446
## 10    35 2002-09-30   503401    467080 12625         492046            0.0231
```

```
# 4. Export the dataframe of Q 3.3 to the PostgreSQL database via API
# Delete the existing table if it exists
dbExecute(con, "DROP TABLE IF EXISTS asset_growth_rate_table")
```

```
## [1] 0
```

```
dbWriteTable(con, name = "asset_growth_rate_table", value = bank_data_df_growth_rate, row.names
= FALSE, overwrite = TRUE)

query <- "SELECT * FROM asset_growth_rate_table"
asset_growth_rate_df <- dbGetQuery(con, query)
head(asset_growth_rate_df)
```

```
##   id       date    asset liability   idx previous_asset asset_growth_rate
## 1  9 2002-06-30   361953    332900 20913         348727        0.03792652
## 2  9 2002-09-30   383246    352456 20914         361953        0.05882808
## 3  9 2002-12-31   371812    340365 20911         383246       -0.02983462
## 4 14 2002-06-30 73600000  69200000 27335       68600000        0.07288630
## 5 14 2002-09-30 72800000  68200000 27336       73600000       -0.01086957
## 6 14 2002-12-31 79600000  74500000 27333       72800000        0.09340659
```

```
# Close the database connection
dbDisconnect(con)
```