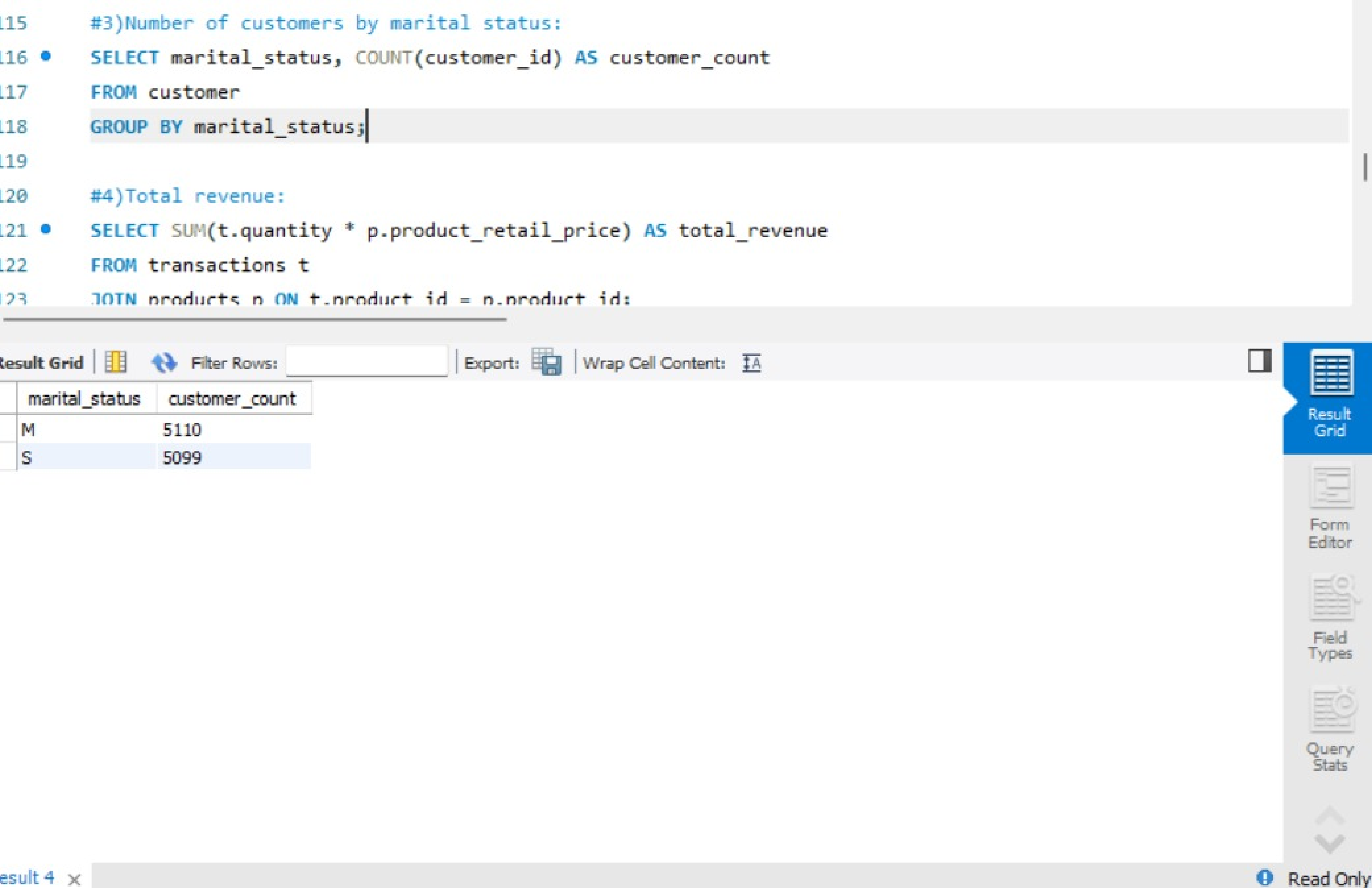
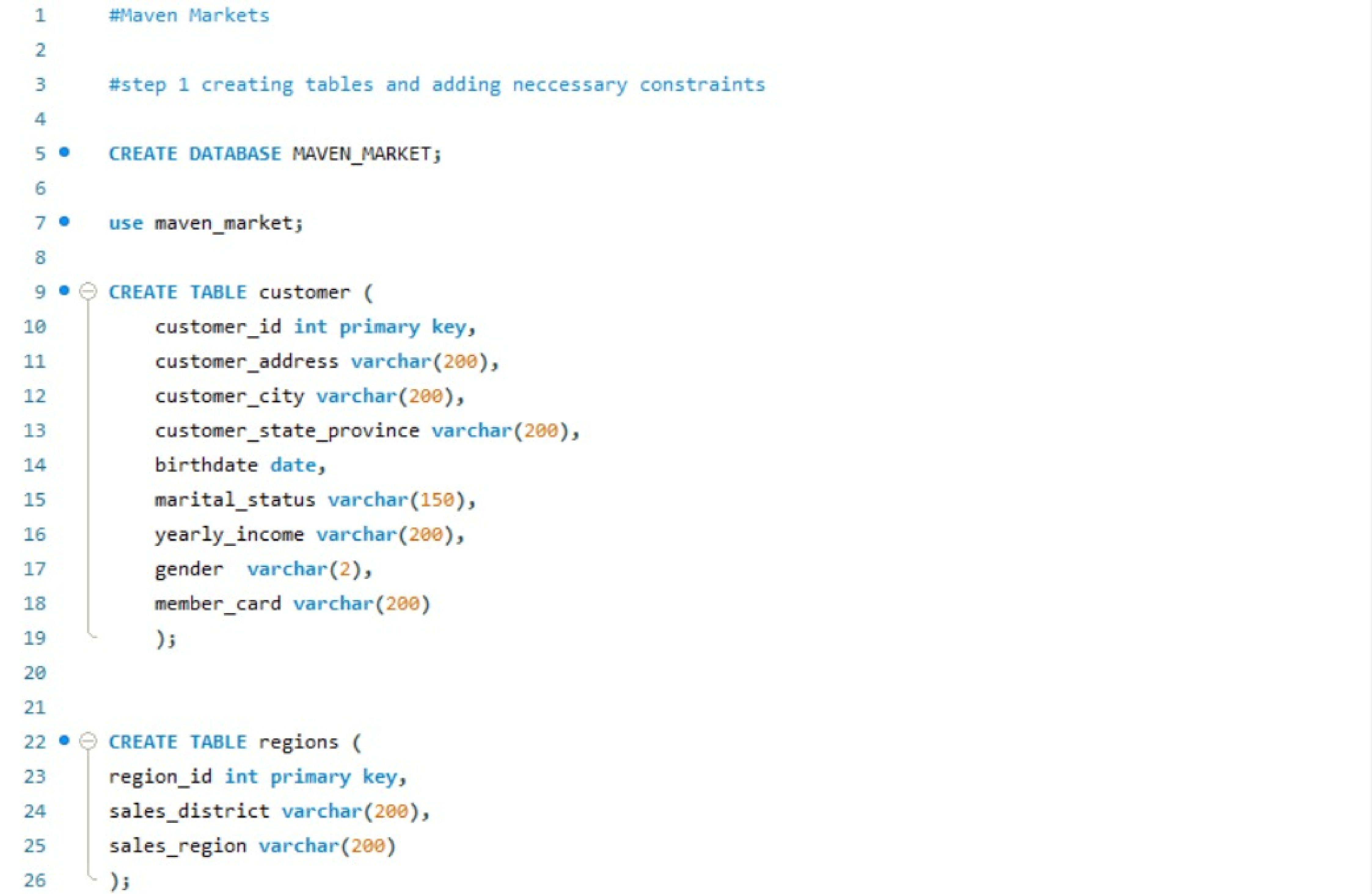
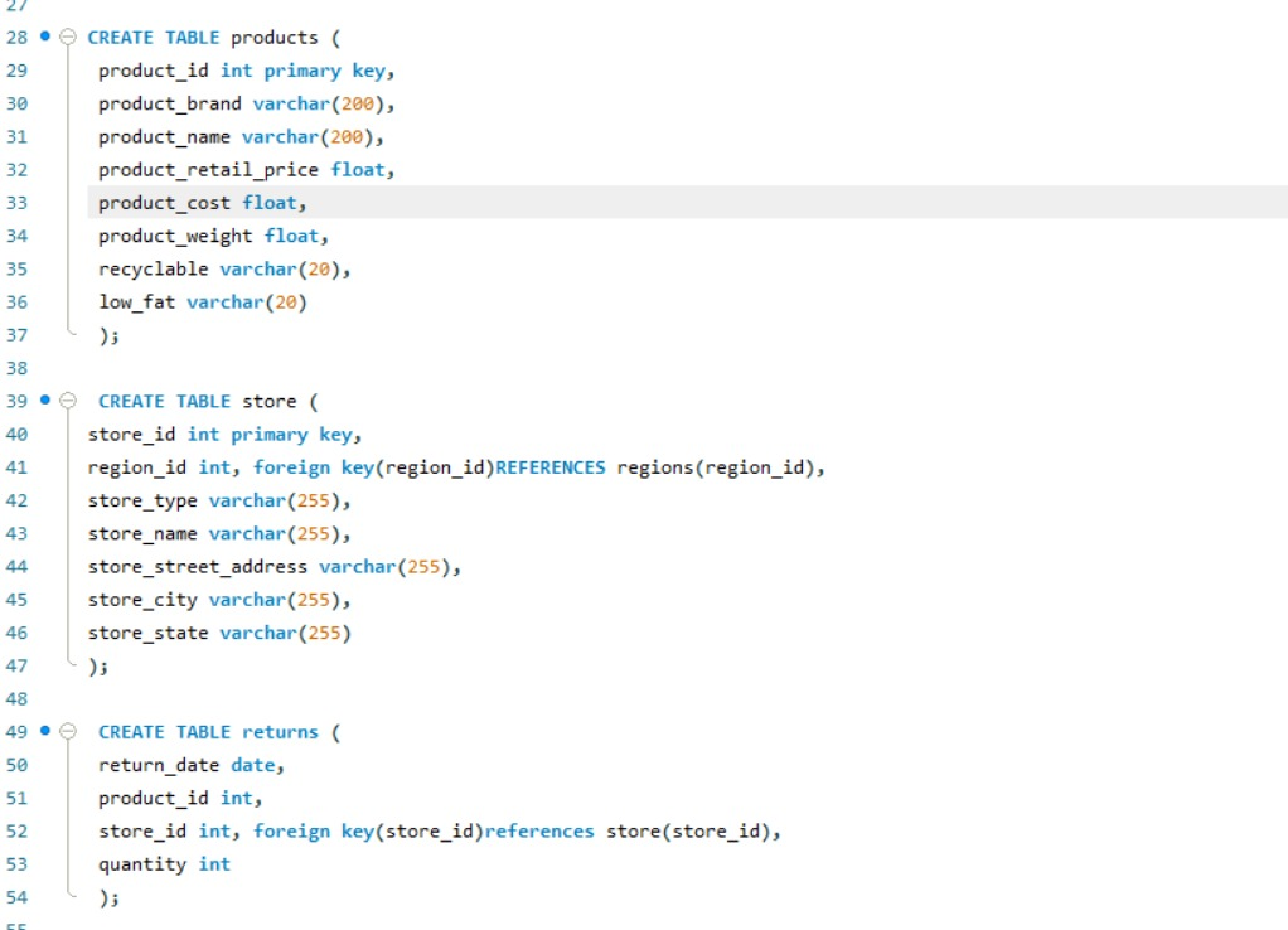


```
#3)Number of customers by marital status:
SELECT marital_status, COUNT(customer_id) AS customer_count
FROM customer
GROUP BY marital_status;

#4)Total revenue:
SELECT SUM(t.quantity * p.product_retail_price) AS total_revenue
FROM transactions t
JOIN products p ON t.product_id = p.product_id;
```

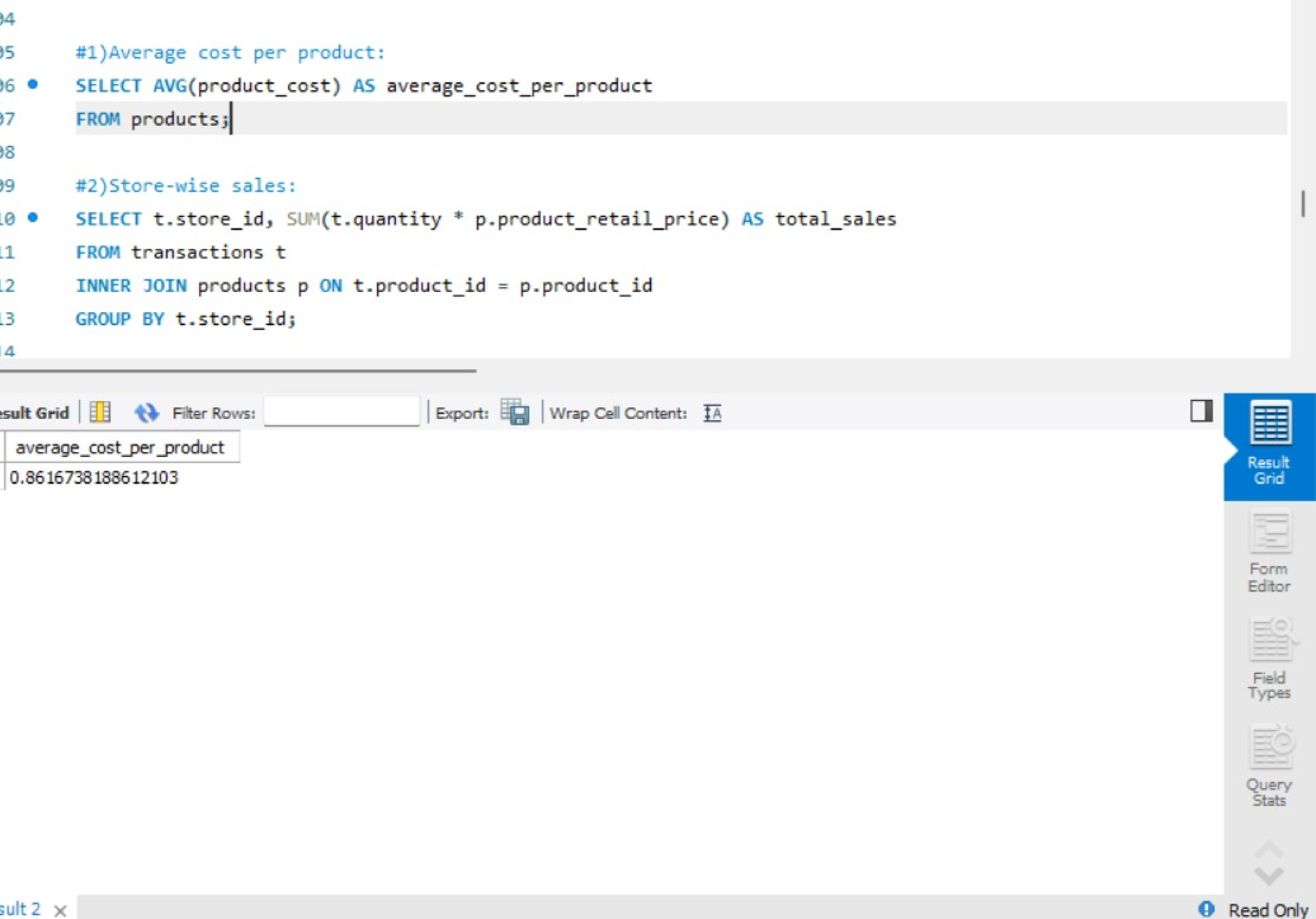
marital_status	customer_count
M	5110
S	5099



```
#Maven Markets

#step 1 creating tables and adding neccessary constraints

CREATE DATABASE MAVEN_MARKET;

use maven_market;

CREATE TABLE customer (
    customer_id int primary key,
    customer_address varchar(200),
    customer_city varchar(200),
    customer_state_province varchar(200),
    birthdate date,
    marital_status varchar(150),
    yearly_income varchar(200),
    gender  varchar(2),
    member_card varchar(200)
    );


CREATE TABLE regions (
region_id int primary key,
sales_district varchar(200),
sales_region varchar(200)
);
```

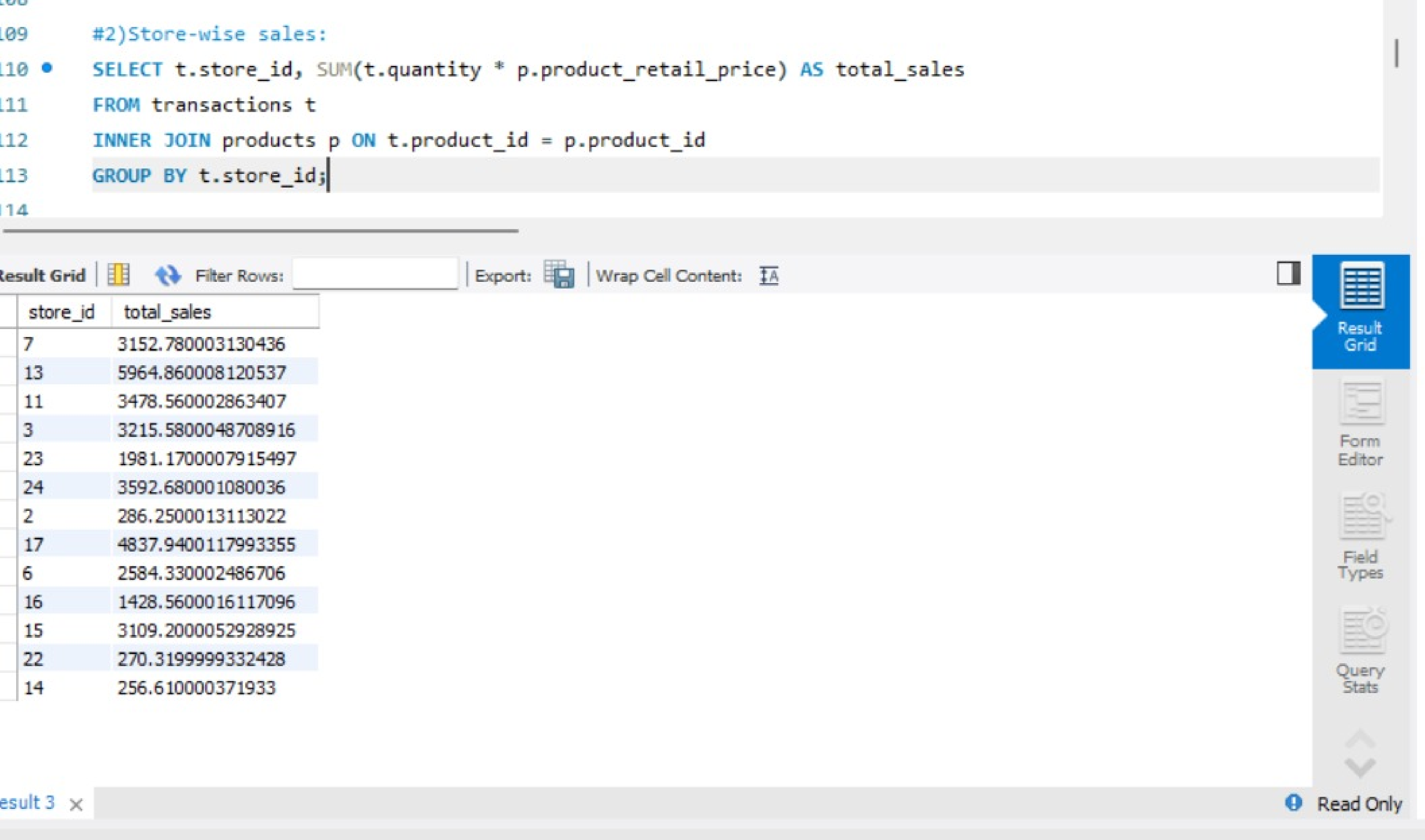


```
CREATE TABLE products (
 product_id int primary key,
 product_brand varchar(200),
 product_name varchar(200),
 product_retail_price float,
 product_cost float,
 product_weight float,
 recyclable varchar(20),
 low_fat varchar(20)
 );

 CREATE TABLE store (
store_id int primary key,
region_id int, foreign key(region_id)REFERENCES regions(region_id),
store_type varchar(255),
store_name varchar(255),
store_street_address varchar(255),
store_city varchar(255),
store_state varchar(255)
);

 CREATE TABLE returns (
 return_date date,
 product_id int,
 store_id int, foreign key(store_id)references store(store_id),
 quantity int
 );
```



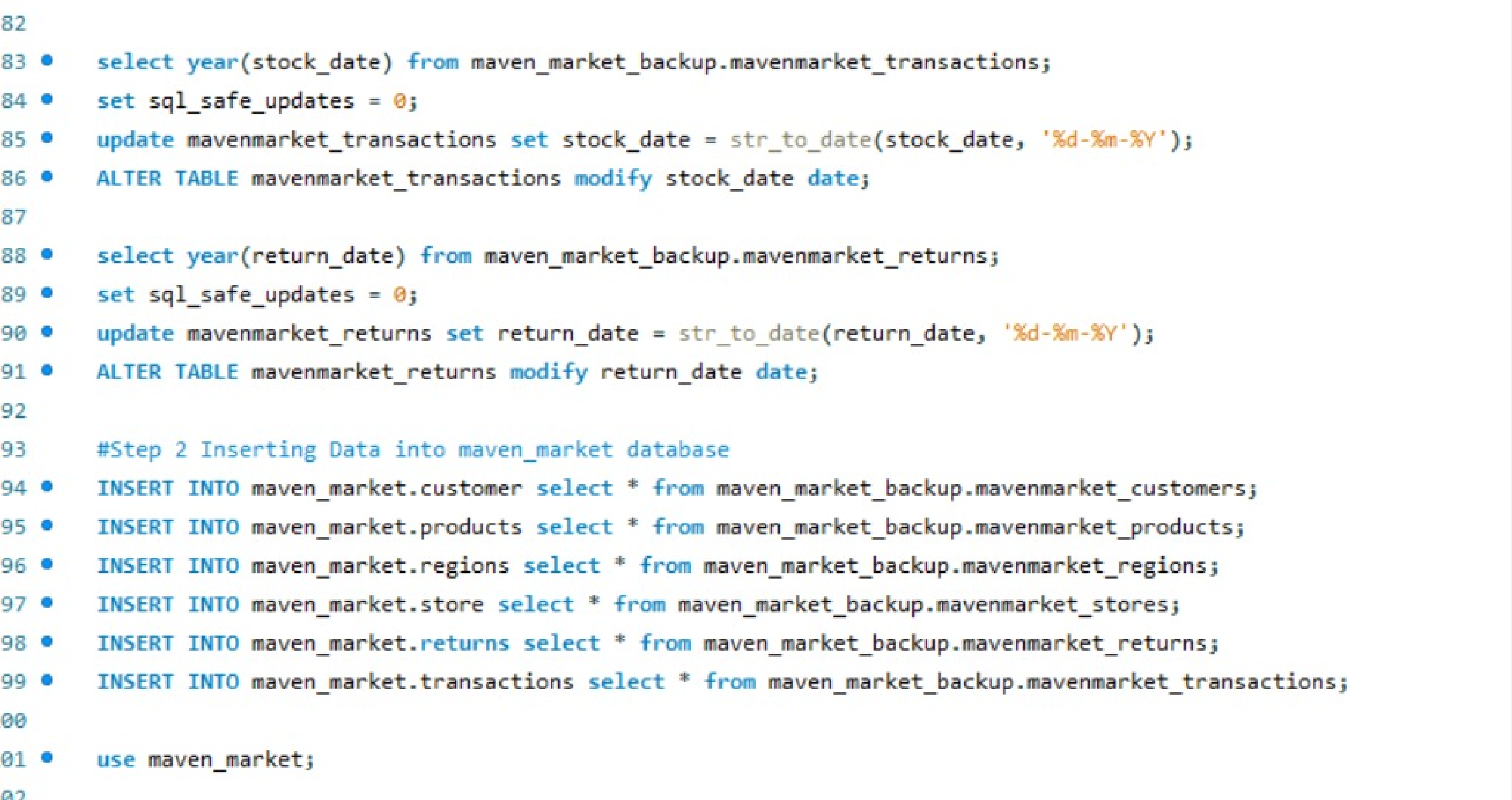
```
#1)Average cost per product:
SELECT AVG(product_cost) AS average_cost_per_product
FROM products;

#2)Store-wise sales:
SELECT t.store_id, SUM(t.quantity * p.product_retail_price) AS total_sales
FROM transactions t
INNER JOIN products p ON t.product_id = p.product_id
GROUP BY t.store_id;
```

average_cost_per_product
0.8616738188612103

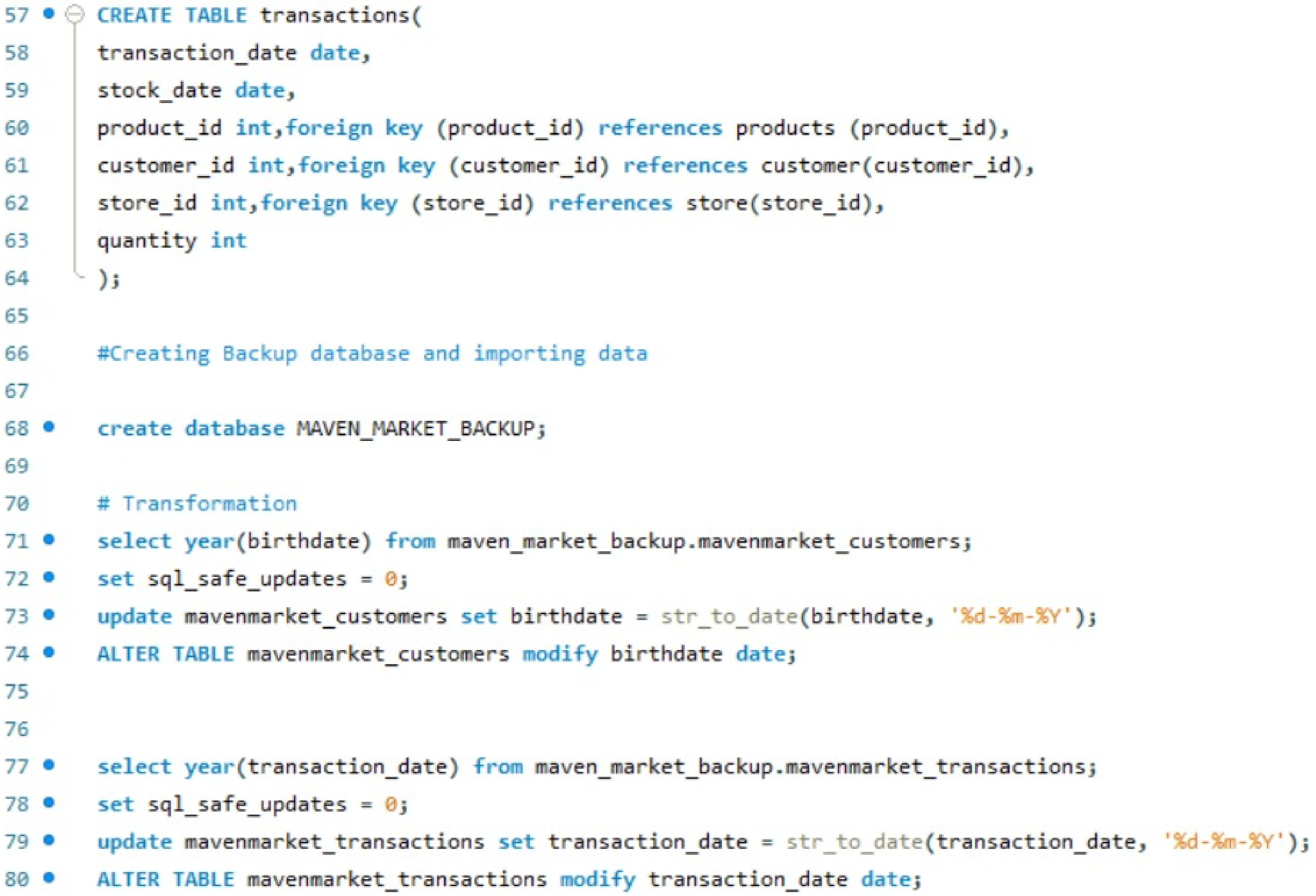


```
#2)Store-wise sales:
SELECT t.store_id, SUM(t.quantity * p.product_retail_price) AS total_sales
FROM transactions t
INNER JOIN products p ON t.product_id = p.product_id
GROUP BY t.store_id;
```

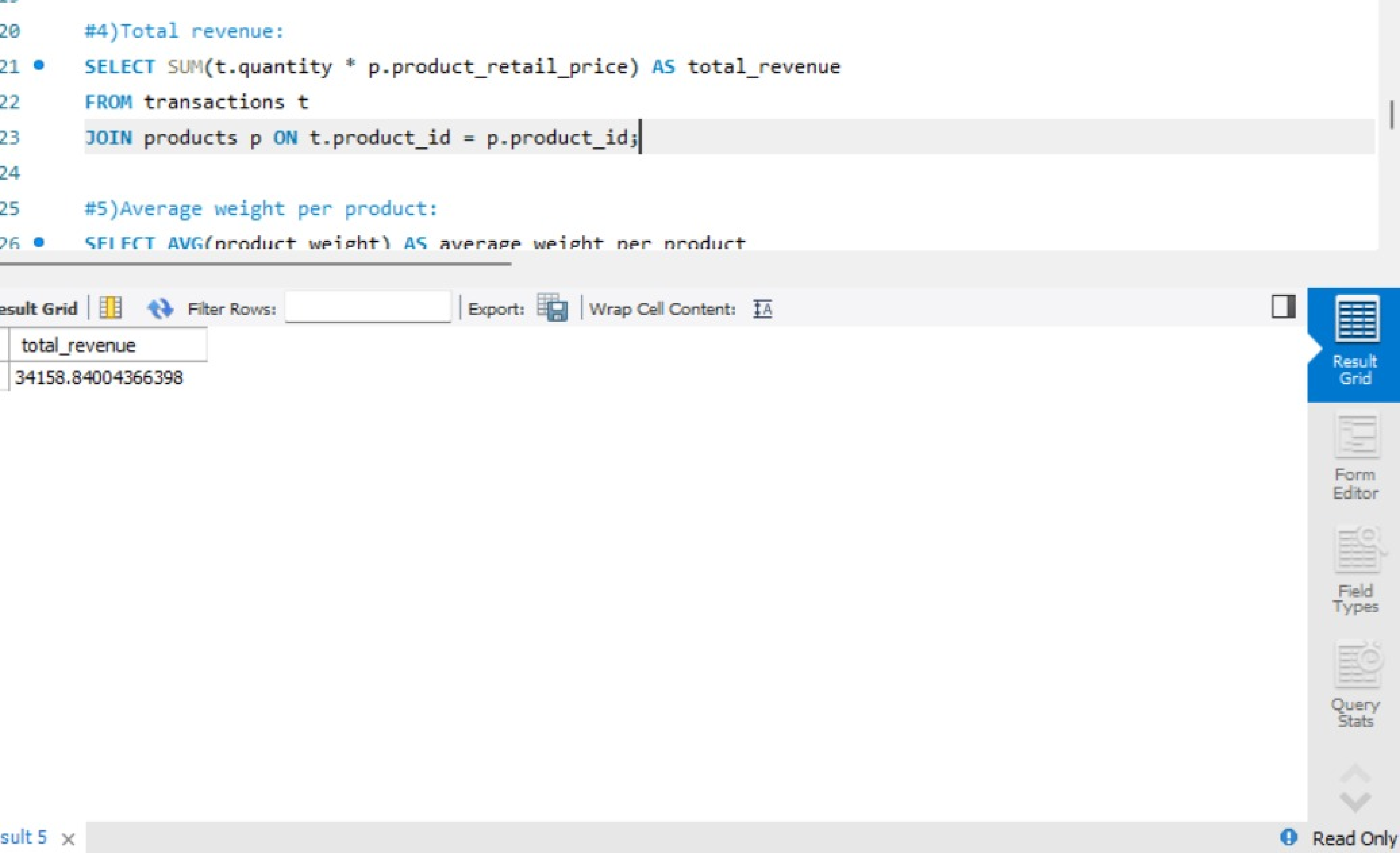
store_id	total_sales
7	3152.780003130436
13	5964.860008120537
11	3478.560002863407
3	3215.5800048708916
23	1981.1700007915497
24	3592.680001080036
2	286.2500013113022
17	4837.9400117993355
6	2584.330002486706
16	1428.5600016117096
15	3109.2000052928925
22	270.3199999332428
14	256.610000371933



```sql

select year(stock_date) from maven_market_backup.mavenmarket_transactions;
set sql_safe_updates = 0;
update mavenmarket_transactions set stock_date = str_to_date(stock_date, '%d-%m-%Y');
ALTER TABLE mavenmarket_transactions modify stock_date date;

select year(return_date) from maven_market_backup.mavenmarket_returns;
set sql_safe_updates = 0;
update mavenmarket_returns set return_date = str_to_date(return_date, '%d-%m-%Y');
ALTER TABLE mavenmarket_returns modify return_date date;

#Step 2 Inserting Data into maven_market database
INSERT INTO maven_market.customer select * from maven_market_backup.mavenmarket_customers;
INSERT INTO maven_market.products select * from maven_market_backup.mavenmarket_products;
INSERT INTO maven_market.regions select * from maven_market_backup.mavenmarket_regions;
INSERT INTO maven_market.store select * from maven_market_backup.mavenmarket_stores;
INSERT INTO maven_market.returns select * from maven_market_backup.mavenmarket_returns;
INSERT INTO maven_market.transactions select * from maven_market_backup.mavenmarket_transactions;

use maven_market;

```

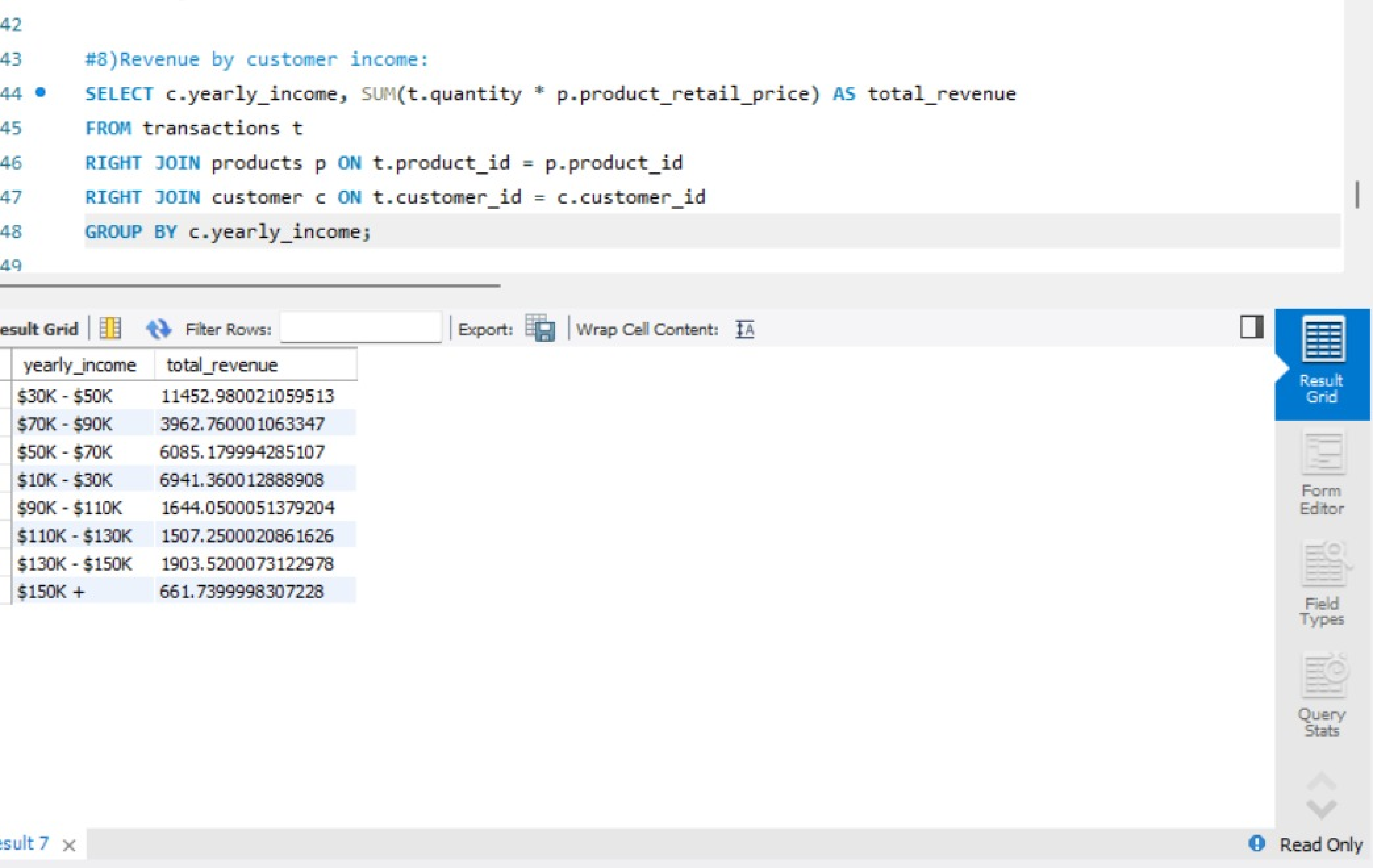


```sql
CREATE TABLE transactions(
transaction_date date,
stock_date date,
product_id int,foreign key (product_id) references products (product_id),
customer_id int,foreign key (customer_id) references customer(customer_id),
store_id int,foreign key (store_id) references store(store_id),
quantity int
);

#Creating Backup database and importing data

create database MAVEN_MARKET_BACKUP;

# Transformation
select year(birthdate) from maven_market_backup.mavenmarket_customers;
set sql_safe_updates = 0;
update mavenmarket_customers set birthdate = str_to_date(birthdate, '%d-%m-%Y');
ALTER TABLE mavenmarket_customers modify birthdate date;


select year(transaction_date) from maven_market_backup.mavenmarket_transactions;
set sql_safe_updates = 0;
update mavenmarket_transactions set transaction_date = str_to_date(transaction_date, '%d-%m-%Y');
ALTER TABLE mavenmarket_transactions modify transaction_date date;
```



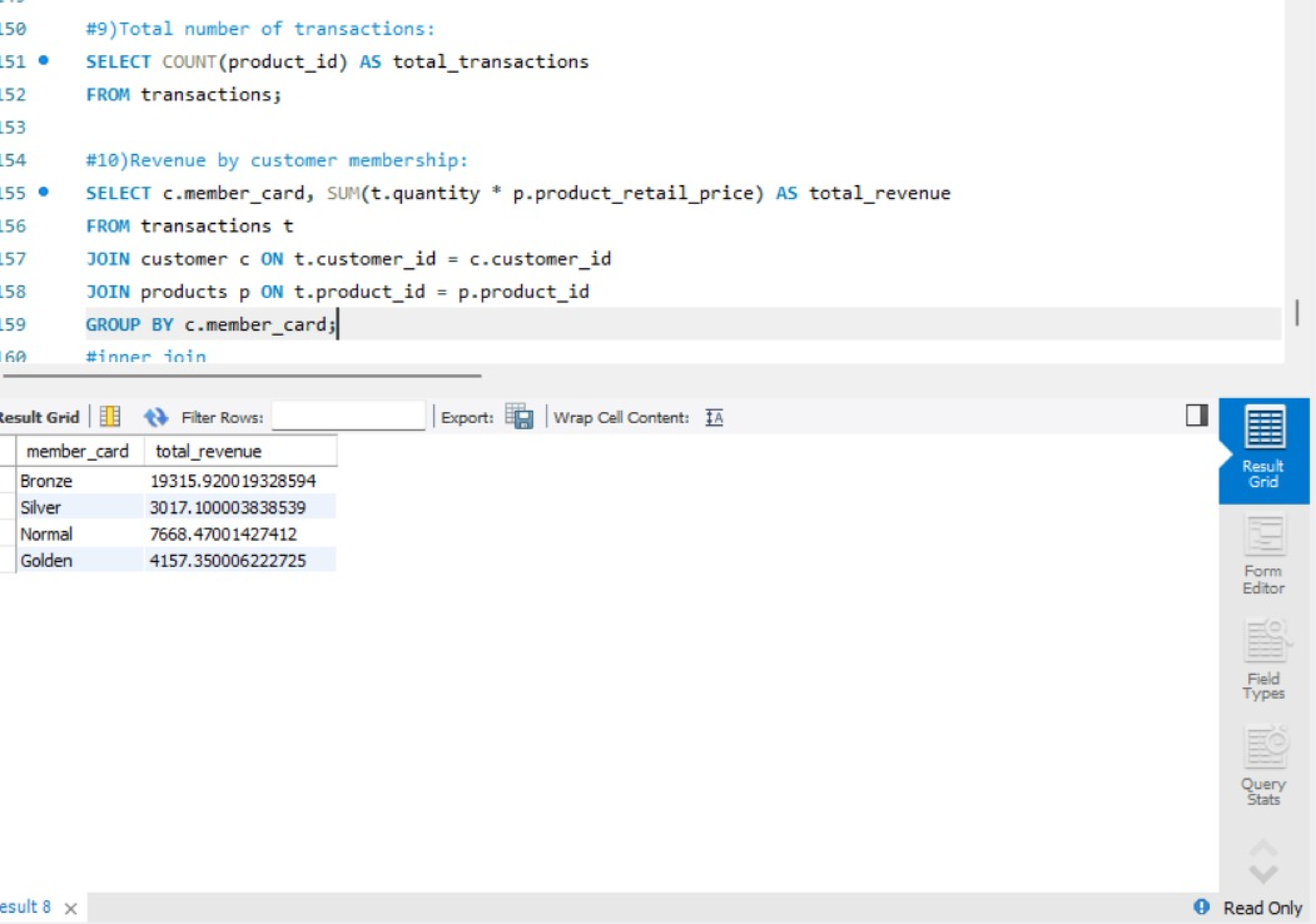
```
#4)Total revenue:
SELECT SUM(t.quantity * p.product_retail_price) AS total_revenue
FROM transactions t
JOIN products p ON t.product_id = p.product_id;

#5)Average weight per product:
SELECT AVG(product_weight) AS average_weight_per_product
```

```
total_revenue
34158.84004366398
```



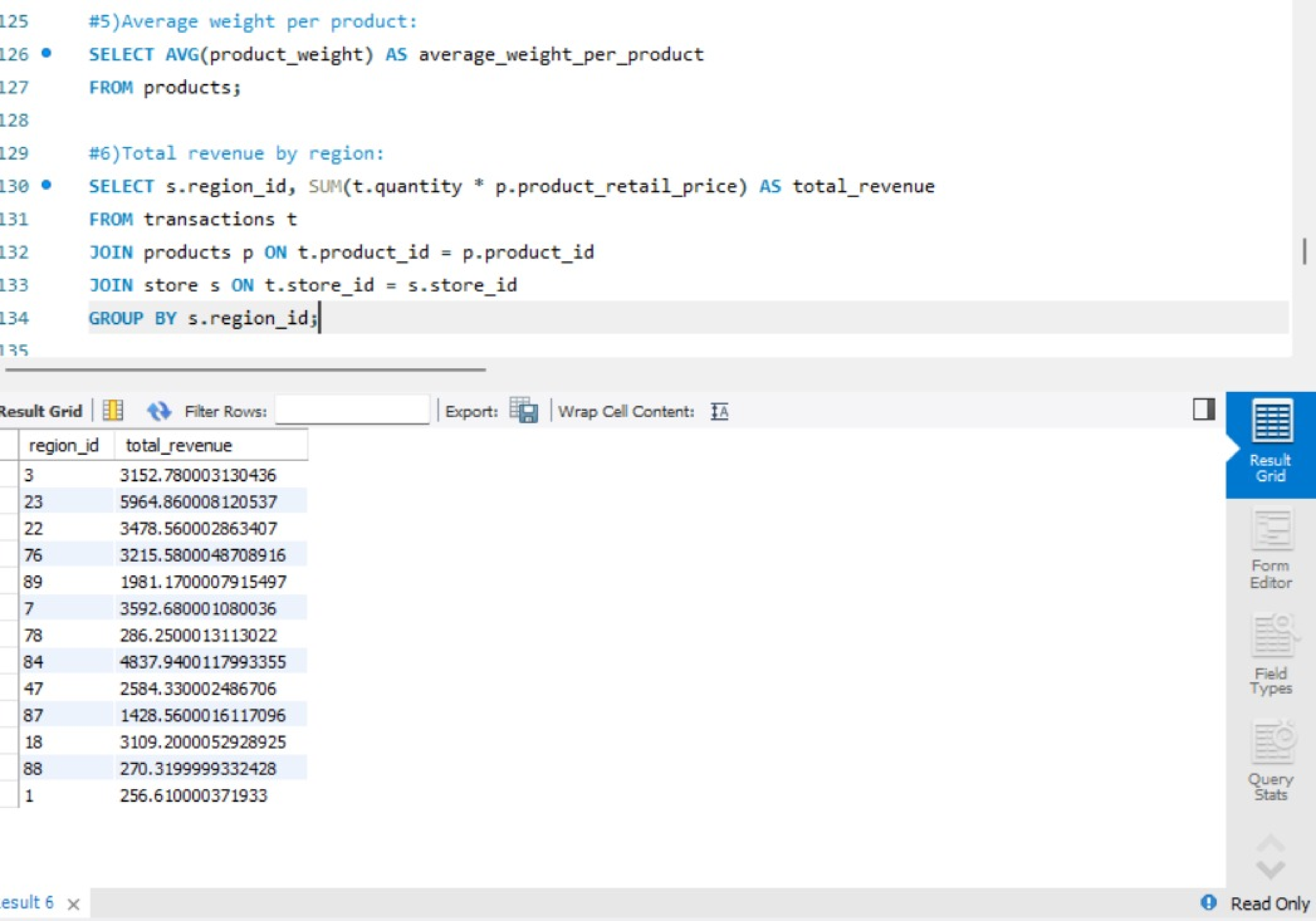
```
#8)Revenue by customer income:
SELECT c.yearly_income, SUM(t.quantity * p.product_retail_price) AS total_revenue
FROM transactions t
RIGHT JOIN products p ON t.product_id = p.product_id
RIGHT JOIN customer c ON t.customer_id = c.customer_id
GROUP BY c.yearly_income;
```

yearly_income	total_revenue
$30K - $50K	11452.980021059513
$70K - $90K	3962.760001063347
$50K - $70K	6085.179994285107
$10K - $30K	6941.360012888908
$90K - $110K	1644.0500051379204
$110K - $130K	1507.2500020861626
$130K - $150K	1903.5200073122978
$150K +	661.7399998307228



```
#9)Total number of transactions:
SELECT COUNT(product_id) AS total_transactions
FROM transactions;

#10)Revenue by customer membership:
SELECT c.member_card, SUM(t.quantity * p.product_retail_price) AS total_revenue
FROM transactions t
JOIN customer c ON t.customer_id = c.customer_id
JOIN products p ON t.product_id = p.product_id
GROUP BY c.member_card;
#inner join
```

member_card	total_revenue
Bronze	19315.920019328594
Silver	3017.100003838539
Normal	7668.47001427412
Golden	4157.350006222725



```
#5)Average weight per product:
SELECT AVG(product_weight) AS average_weight_per_product
FROM products;

#6)Total revenue by region:
SELECT s.region_id, SUM(t.quantity * p.product_retail_price) AS total_revenue
FROM transactions t
JOIN products p ON t.product_id = p.product_id
JOIN store s ON t.store_id = s.store_id
GROUP BY s.region_id;
```

region_id	total_revenue
3	3152.780003130436
23	5964.860008120537
22	3478.560002863407
76	3215.5800048708916
89	1981.1700007915497
7	3592.680001080036
78	286.2500013113022
84	4837.9400117993355
47	2584.330002486706
87	1428.5600016117096
18	3109.2000052928925
88	270.3199999332428
1	256.610000371933