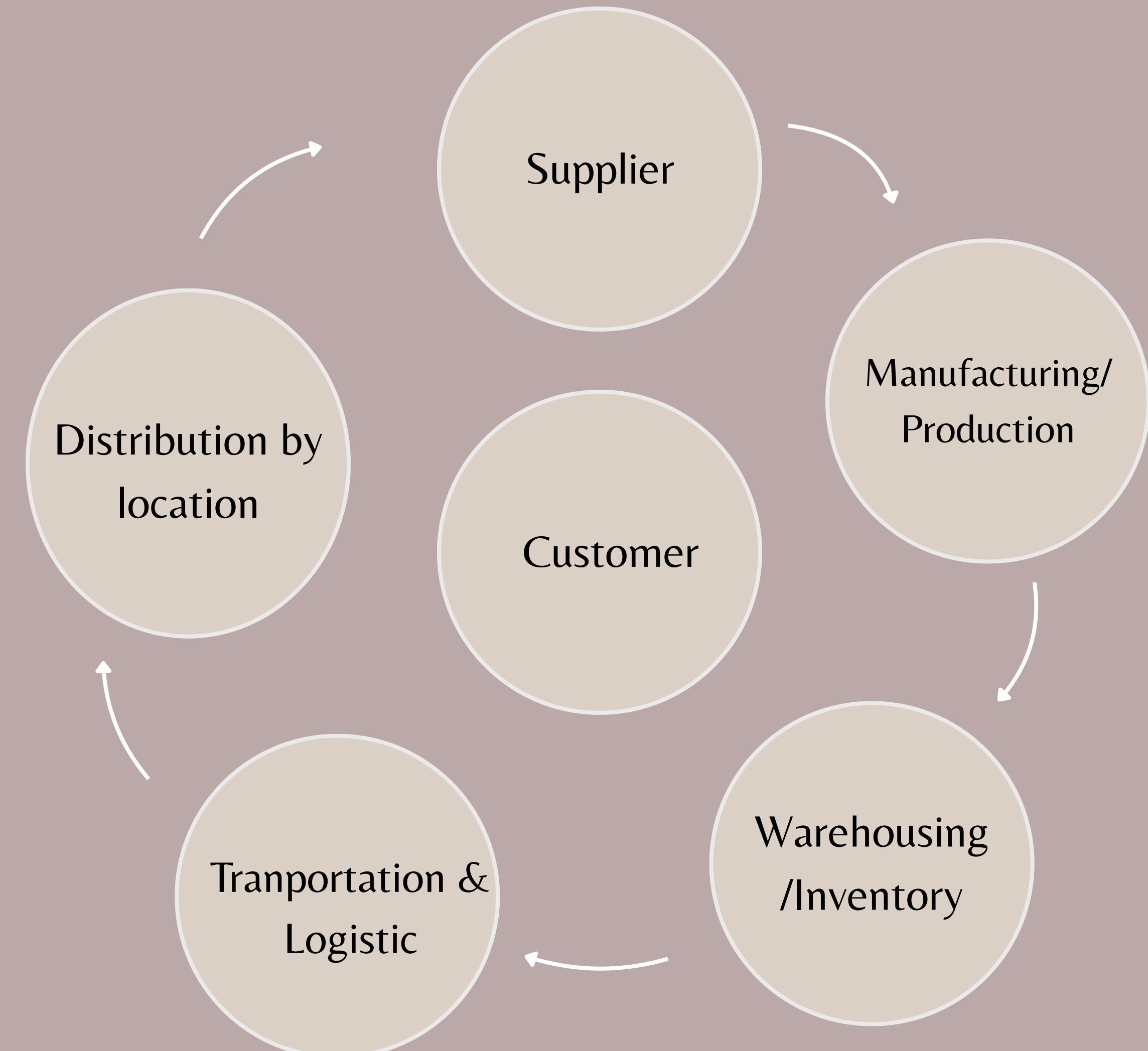


by Monali

Supply Chain Data Project

Skincare, Haircare
& Cosmetics



INTRODUCTION

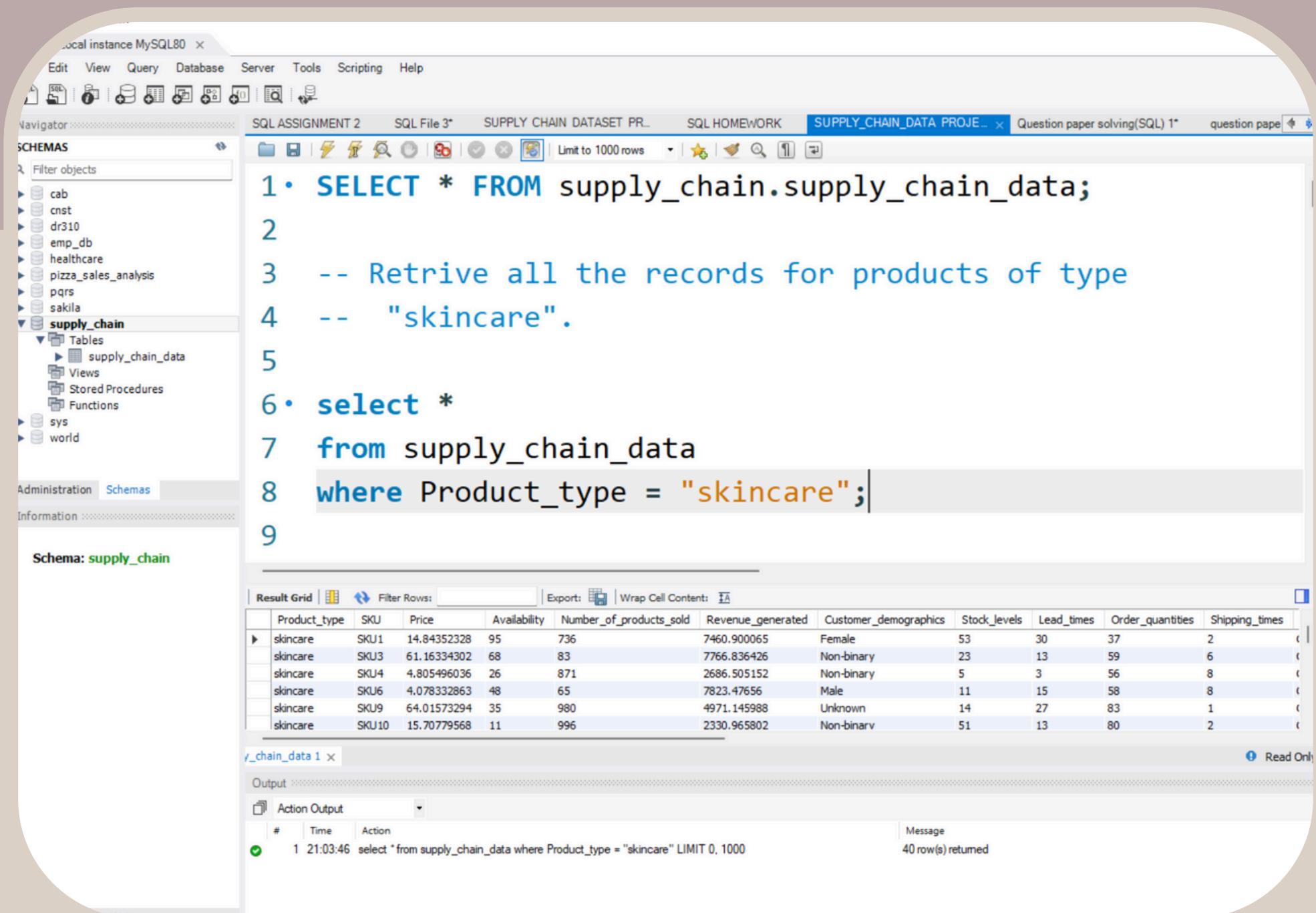
- The project focuses on Supply Chain Data Analysis using SQL.
- Aims to extract actionable insights from raw supply chain data.
- Covers multiple aspects like:
Product analysis (types, pricing, availability).
- Customer insights (demographics & location).
- Supplier performance (costs, lead times, defect rates).

OBJECTIVES

- To analyze product performance by retrieving and filtering records based on product type, price, and availability.
- To understand customer demographics and location patterns for better market segmentation and targeting.
- To measure key supply chain metrics such as total SKUs, defect rates, and manufacturing lead times.
- To evaluate supplier efficiency by comparing costs, lead times, and defect rates across suppliers.
- To assess financial performance by calculating revenue generated per product, location, and transportation mode.
- To identify operational bottlenecks by analyzing production volumes, defect rates, and transportation costs.
- To support decision-making in procurement, logistics, and customer service through data-driven insights.

PROBLEM STATEMENTS

- Retrieve all the records for products of type "skincare".



The screenshot shows the MySQL Workbench interface with a query editor and a results grid.

Query Editor:

```
1• SELECT * FROM supply_chain.supply_chain_data;
2
3 -- Retrieve all the records for products of type
4 -- "skincare".
5
6• select *
7 from supply_chain_data
8 where Product_type = "skincare";
```

Results Grid:

Product_type	SKU	Price	Availability	Number_of_products_sold	Revenue_generated	Customer_demographics	Stock_levels	Lead_times	Order_quantities	Shipping_times
skincare	SKU1	14.84352328	95	736	7460.900065	Female	53	30	37	2
skincare	SKU3	61.16334302	68	83	7766.836426	Non-binary	23	13	59	6
skincare	SKU4	4.805496036	26	871	2686.505152	Non-binary	5	3	56	8
skincare	SKU6	4.078332863	48	65	7823.47656	Male	11	15	58	8
skincare	SKU9	64.01573294	35	980	4971.145988	Unknown	14	27	83	1
skincare	SKU10	15.70779568	11	996	2330.965802	Non-binary	51	13	80	2

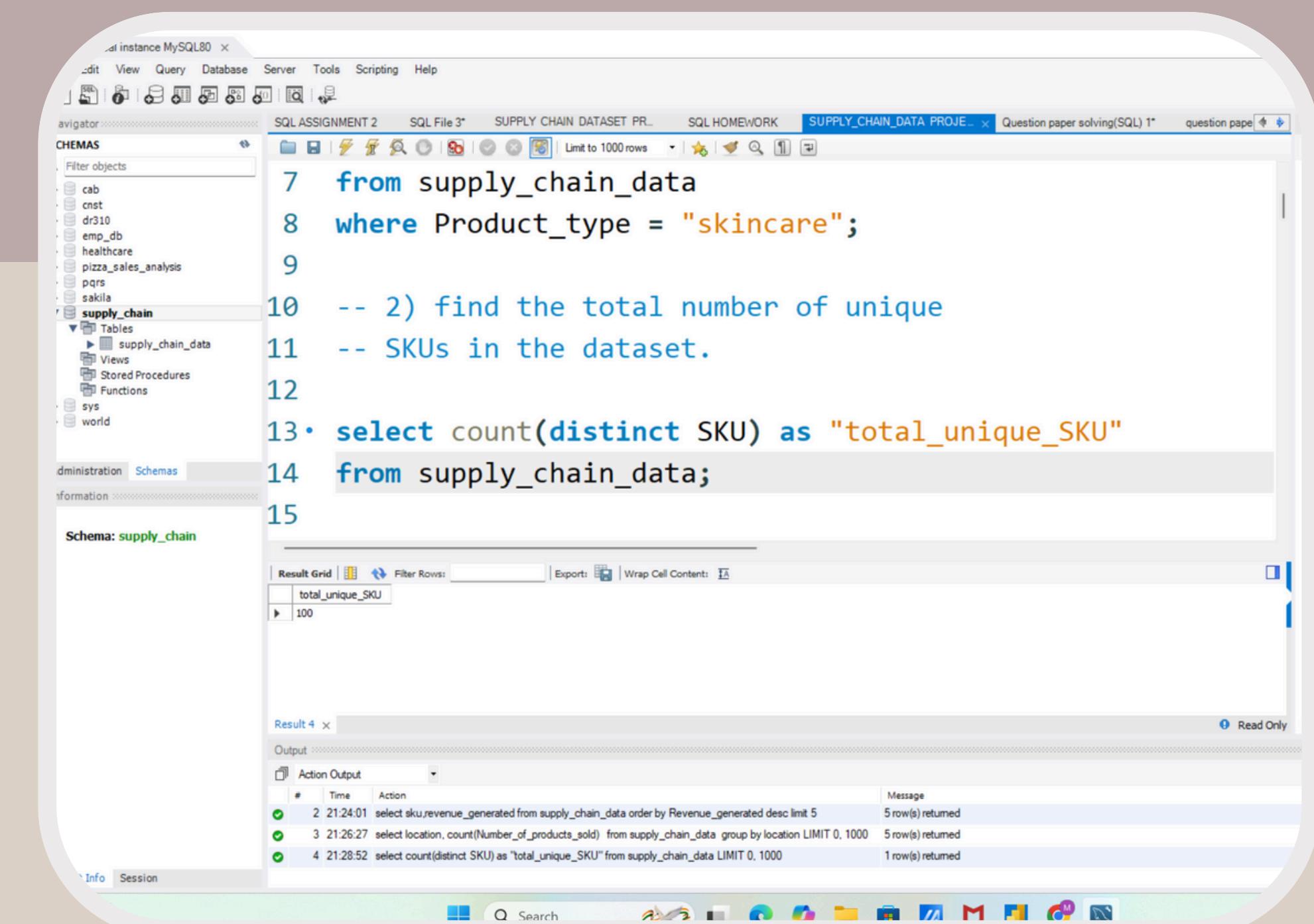
Action Output:

#	Time	Action
1	21:03:46	select * from supply_chain_data where Product_type = "skincare" LIMIT 0, 1000

Message: 40 row(s) returned

PROBLEM STATEMENT 2

- find the total number of unique SKUs in the dataset.



The screenshot shows the MySQL Workbench interface with a query editor window open. The code entered is:

```
7  from supply_chain_data
8  where Product_type = "skincare";
9
10 -- 2) find the total number of unique
11 -- SKUs in the dataset.
12
13 • select count(distinct SKU) as "total_unique_SKU"
14   from supply_chain_data;
15
```

The result grid shows a single row with the value 100 under the column 'total_unique_SKU'.

The 'Result 4' tab shows the history of actions taken:

#	Time	Action	Message
2	21:24:01	select sku,revenue_generated from supply_chain_data order by Revenue_generated desc limit 5	5 row(s) returned
3	21:26:27	select location, count(Number_of_products_sold) from supply_chain_data group by location LIMIT 0, 1000	5 row(s) returned
4	21:28:52	select count(distinct SKU) as "total_unique_SKU" from supply_chain_data LIMIT 0, 1000	1 row(s) returned

PROBLEM STATEMENTS 3

- list products where price is greater than 50

The screenshot shows the MySQL Workbench interface with a query editor and a result grid.

Query Editor:

```
15
16  -- list products where price is greater than 50
17
18 • select *
19   from supply_chain_data
20  where price > 50;
21
```

Result Grid:

Product_type	SKU	Price	Availability	Number_of_products_sold	Revenue_generated	Customer_demographics	Stock_levels	Lead_times	Order_quantities	Shipping_times
haircare	SKU0	69.80800554	55	802	8661.996792	Non-binary	58	7	96	4
skincare	SKU3	61.16334302	68	83	7766.836426	Non-binary	23	13	59	6
cosmetics	SKU8	68.71759675	78	150	7517.363211	Female	5	10	15	7
skincare	SKU9	64.01573294	35	980	4971.145988	Unknown	14	27	83	1
skincare	SKU11	90.63545998	95	960	6099.944116	Female	46	23	60	1
haircare	SKU12	71.21338908	41	336	2873.741446	Unknown	100	30	85	4
skincare	SKU14	99.17132864	26	562	8653.570926	Non-binary	54	29	78	5
cosmetics	SKU17	81.46253437	82	126	2629.396435	Female	45	17	85	9
skincare	SKU19	51.12387009	100	187	2553.495585	Unknown	48	11	94	3
skincare	SKU20	96.34107244	22	320	8128.027697	Unknown	27	12	68	6
cosmetics	SKU21	84.89386898	60	601	7087.052696	Unknown	69	25	7	6
haircare	SKU26	97.44694662	9	353	3716.493326	Male	59	16	48	4
cosmetics	SKU27	92.55736081	42	352	2686.457224	Unknown	47	9	62	8
cosmetics	SKU29	63.44755919	3	253	8318.903195	Female	45	5	67	7

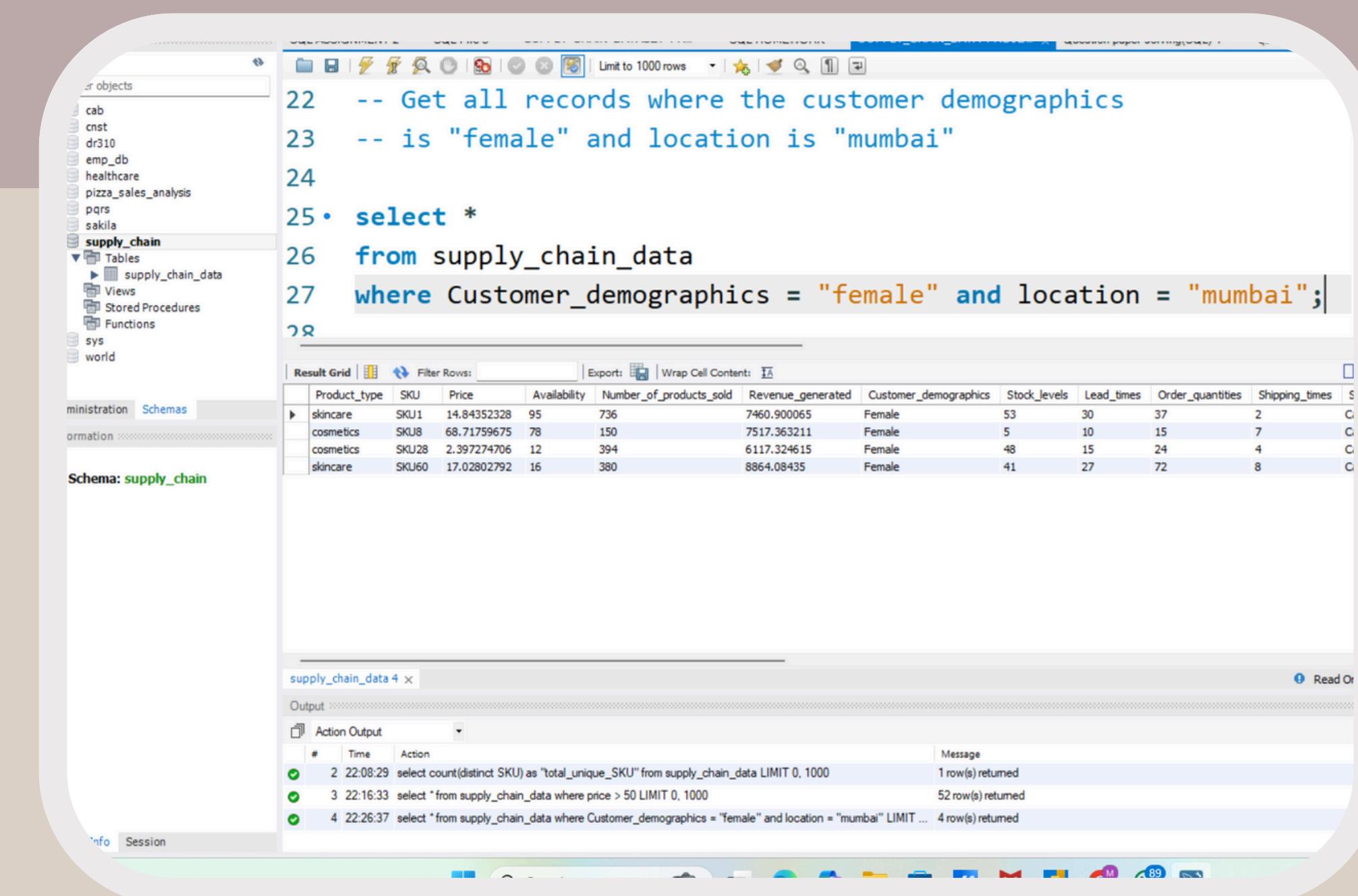
Action Output:

#	Time	Action	Message
1	21:51:47	select * from supply_chain_data where Product_type = "skincare" LIMIT 0, 1000	40 row(s) returned
2	22:08:29	select count(distinct SKU) as "total_unique_SKU" from supply_chain_data LIMIT 0, 1000	1 row(s) returned
3	22:16:33	select * from supply_chain_data where price > 50 LIMIT 0, 1000	52 row(s) returned

PROBLEM STATEMENTS 4

-

Get all records where the customer demographics is "female" and location is "mumbai"



The screenshot shows a MySQL Workbench interface with the following details:

- Schemas:** The left sidebar shows the database structure with the schema **supply_chain** selected.
- Query Editor:** The main area contains the following SQL code:

```
22 -- Get all records where the customer demographics
23 -- is "female" and location is "mumbai"
24
25 • select *
26 from supply_chain_data
27 where Customer_demographics = "female" and location = "mumbai";
```
- Result Grid:** Below the code, the result grid displays the following data:

Product_type	SKU	Price	Availability	Number_of_products_sold	Revenue_generated	Customer_demographics	Stock_levels	Lead_times	Order_quantities	Shipping_times	Created_at
skincare	SKU1	14.84352328	95	736	7460.900065	Female	53	30	37	2	2023-09-22 10:00:00
cosmetics	SKU8	68.71759675	78	150	7517.363211	Female	5	10	15	7	2023-09-22 10:00:00
cosmetics	SKU28	2.397274706	12	394	6117.324615	Female	48	15	24	4	2023-09-22 10:00:00
skincare	SKU60	17.02802792	16	380	8864.08435	Female	41	27	72	8	2023-09-22 10:00:00

- Action Output:** At the bottom, the action output shows the execution history:

#	Time	Action	Message
2	22:08:29	select count(distinct SKU) as "total_unique_SKU" from supply_chain_data LIMIT 0, 1000	1 row(s) returned
3	22:16:33	select * from supply_chain_data where price > 50 LIMIT 0, 1000	52 row(s) returned
4	22:26:37	select * from supply_chain_data where Customer_demographics = "female" and location = "mumbai" LIMIT 0, 1000	4 row(s) returned

Problem statement 5

- list all the products whose name starts with "skin"

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** A tree view showing databases like cab, cnst, dr310, emp_db, healthcare, pizza_sales_analysis, pqrs, sakila, and supply_chain.
- Tables:** Under the supply_chain database, there is a table named supply_chain_data.
- Query Editor:** The SQL tab contains the following code:

```
127
128 -- list all the products whose name starts with "skin"
129 • select product_type,price
130 from supply_chain_data
131 where Product_type like 'skin%';
132
```
- Result Grid:** The results of the query are displayed in a grid format. The columns are product_type and price. The data shows multiple rows of skincare products with their respective prices.
- Action Output:** The session history shows three actions:

#	Time	Action	Message
1	23:55:16	select product_type,price from supply_chain_data where product_type like 'cream' and 'care' LIMIT 0, 1000	0 row(s) returned
2	00:00:59	select productname,price from products where productname like 'skin' LIMIT 0, 1000	Error Code: 1146. Table 'supply_chain.products' doesn't exist
3	00:02:29	select product_type,price from supply_chain_data where Product_type like 'skin%' LIMIT 0, 1000	40 row(s) returned

PROBLEM STATEMENT 6

- calculate the average price for each product type.

The screenshot shows a MySQL Workbench interface with the following details:

- Object Navigator:** Shows the database schema with the **supply_chain** database selected.
- SQL Editor:** Contains the following SQL code:

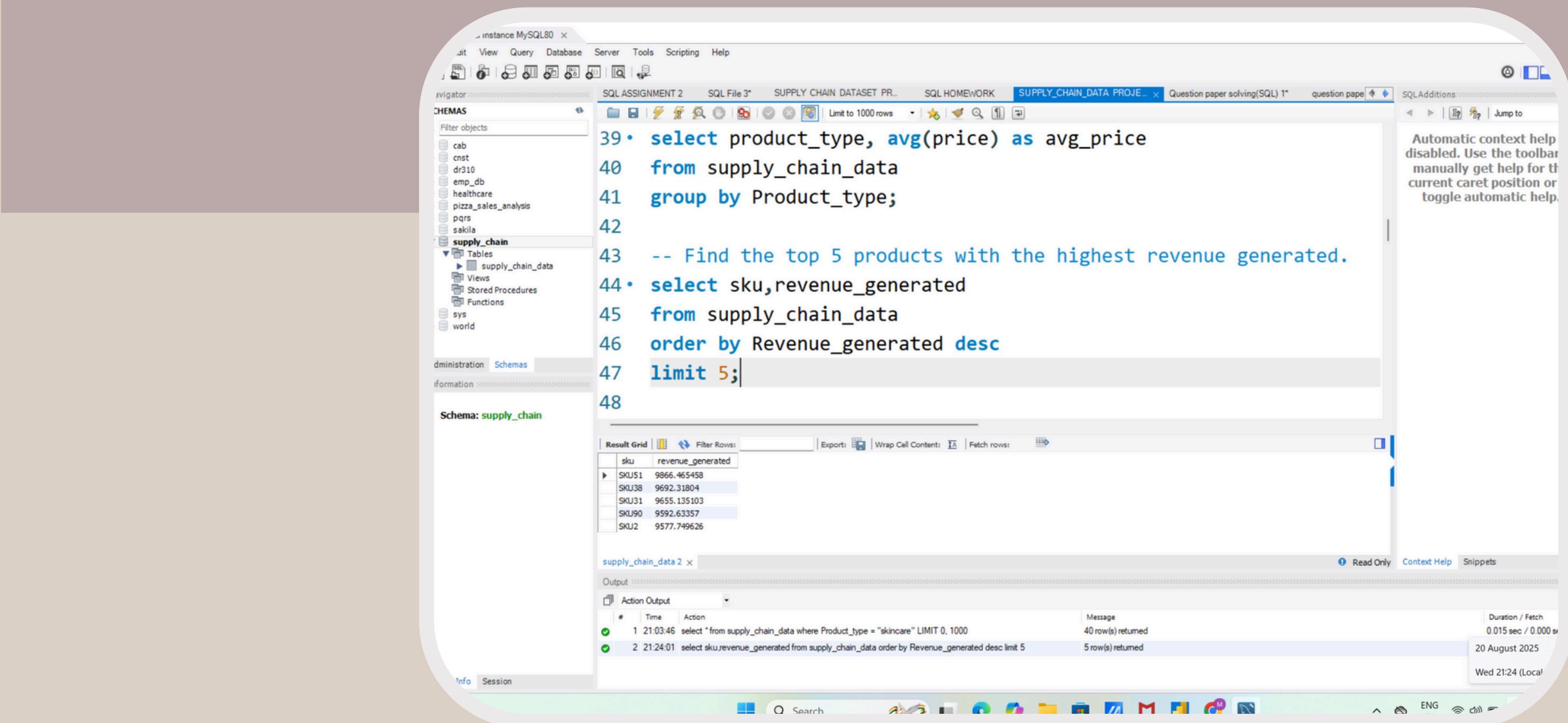
```
36
37  -- calculate the average price for each product type.
38
39 • select product_type, avg(price) as avg_price
40   from supply_chain_data
41   group by Product_type;
```
- Result Grid:** Displays the results of the query:

product_type	avg_price
haircare	46.01427887361765
skincare	47.25932887995002
cosmetics	57.36105759961537
- Output Window:** Shows the execution history:

#	Time	Action	Message
4	22:26:37	select *from supply_chain_data where Customer_demographics = "female" and location = "mumbai" LIMIT ...	4 row(s) returned
5	22:35:46	select *from supply_chain_data where Availability > 80 LIMIT 0, 1000	23 row(s) returned
6	22:41:11	select product_type, avg(price) as avg_price from supply_chain_data group by Product_type LIMIT 0, 1000	3 row(s) returned

Problem statement 7

- Find the top 5 products with the highest revenue generated.



The screenshot shows a MySQL Workbench interface with the following details:

- Schemas:** supply_chain
- Query Editor:** Contains two SQL scripts:
 - Script 1 (lines 39-42):

```
39 • select product_type, avg(price) as avg_price
40   from supply_chain_data
41 group by Product_type;
```
 - Script 2 (lines 43-48):

```
43 -- Find the top 5 products with the highest revenue generated.
44 • select sku,revenue_generated
45   from supply_chain_data
46 order by Revenue_generated desc
47 limit 5;
```
- Result Grid:** Displays the results of the second query:

sku	revenue_generated
SKU51	9866.465458
SKU38	9692.31804
SKU31	9655.135103
SKU90	9592.63357
SKU2	9577.749626
- Action Output:** Shows the execution history:

#	Time	Action	Message	Duration / Fetch
1	21:03:46	select * from supply_chain_data where Product_type = "skincare" LIMIT 0, 1000	40 row(s) returned	0.015 sec / 0.000 sec
2	21:24:01	select sku,revenue_generated from supply_chain_data order by Revenue_generated desc limit 5	5 row(s) returned	20 August 2025

Problem statement 8

- Replace "Pending" with "In-Process" in inspection results

The screenshot shows the MySQL Workbench interface with the following details:

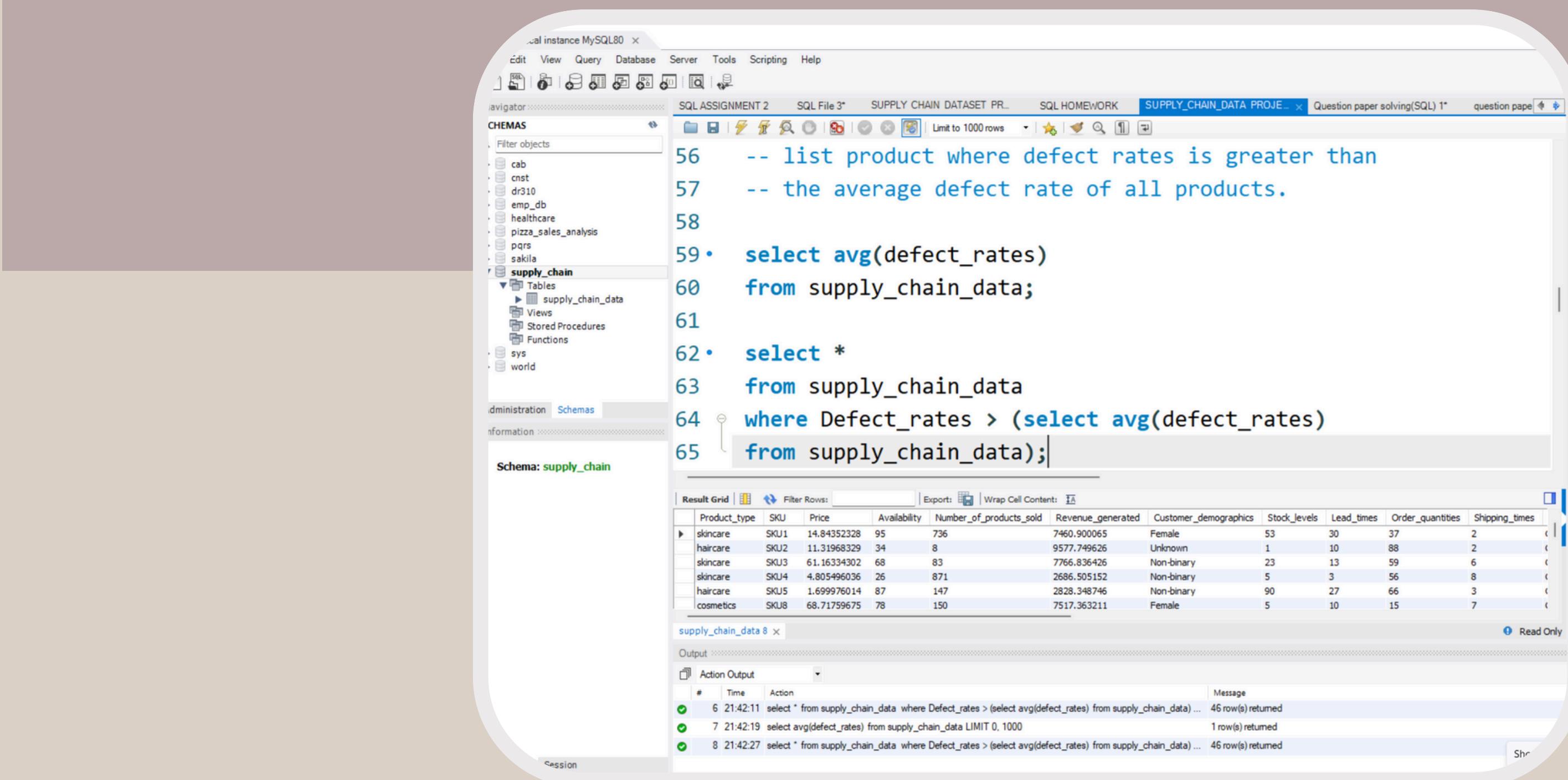
- Top Bar:** Local instance MySQL80 x, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Includes icons for New Connection, Open Connection, Save, Print, Copy, Paste, Find, Replace, and others.
- Navigation:** Shows the current connection as Local instance MySQL80 x and various tabs like SQL ASSIGNMENT 2, SQL File 3*, SUPPLY CHAIN DATASET PR..., SQL HOMEWORK, SUPPLY_CHAIN_DATA PROJEC..., Question paper solving(SQL) 1*, question paper solving, and SQLAdditions.
- Schemas:** A tree view of schemas including cab, cnst, dr310, emp_db, healthcare, pizza_sales_analysis, pqrs, sakila, supply_chain (Tables: supply_chain_data, Views, Stored Procedures, Functions), sys, and world.
- Code Editor:** Displays the following SQL code:

```
131 where Product_type like 'skin%';
132
133 -- REPLACE Query - Clean inspection results
134 • SELECT "SKU",
135         REPLACE("Inspection results", 'Pending', 'In-Process') AS UpdatedInspection
136 FROM supply_chain_data;
137
138 -- Rank products by revenue within each product type.
139 • SELECT Product_Type, Supplier_Name,
140         RANK() OVER (PARTITION BY Product_Type ORDER BY (Price * order_quantities) DE
```
- Result Grid:** Shows the output of the query with columns SKU and UpdatedInspection, containing 10 rows of "SKU Inspection results".
- Action Output:** Shows the execution history with the following entries:

#	Time	Action	Message	Duration / Fetch
8	00:14:17	SELECT Supplier_name, REPLACE(Supplier_Name, 'Ltd', 'Limited') AS UpdatedName FROM supply_chain...	100 row(s) returned	0.000 sec / 0.000 sec
9	00:27:33	SELECT ProductID, REPLACE(ProductType, 'Cosmetics', 'Beauty Products') AS UpdatedProductType FR...	Error Code: 1054. Unknown column 'ProductID' in field list	0.000 sec
10	00:28:02	SELECT "SKU", REPLACE("Inspection results", 'Pending', 'In-Process') AS UpdatedInspection FROM sup...	100 row(s) returned	0.000 sec / 0.000 sec
- Status Bar:** Shows warning, Info, Session, and a system tray with icons for network, battery, volume, and other system status.

Problem statement 9

- list product where defect rates is greater than the average defect rate of all products.



The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Contains the following SQL code:

```
56  -- list product where defect rates is greater than
57  -- the average defect rate of all products.
58
59 • select avg(defect_rates)
60   from supply_chain_data;
61
62 • select *
63   from supply_chain_data
64   where Defect_rates > (select avg(defect_rates)
65     from supply_chain_data);
```
- Results Grid:** Displays a table titled "supply_chain_data" with 8 rows of data. The columns are: Product_type, SKU, Price, Availability, Number_of_products_sold, Revenue_generated, Customer_demographics, Stock_levels, Lead_times, Order_quantities, and Shipping_times. The data is as follows:

Product_type	SKU	Price	Availability	Number_of_products_sold	Revenue_generated	Customer_demographics	Stock_levels	Lead_times	Order_quantities	Shipping_times
skincare	SKU1	14.84352328	95	736	7460.900065	Female	53	30	37	2
haircare	SKU2	11.31968329	34	8	9577.749626	Unknown	1	10	88	2
skincare	SKU3	61.16334302	68	83	7766.836426	Non-binary	23	13	59	6
skincare	SKU4	4.805496036	26	871	2686.505152	Non-binary	5	3	56	8
haircare	SKU5	1.699976014	87	147	2828.348746	Non-binary	90	27	66	3
cosmetics	SKU8	68.71759675	78	150	7517.363211	Female	5	10	15	7

- Action Output:** Shows the execution history with three entries:

#	Time	Action	Message
6	21:42:11	select * from supply_chain_data where Defect_rates > (select avg(defect_rates) from supply_chain_data)	46 row(s) returned
7	21:42:19	select avg(defect_rates) from supply_chain_data LIMIT 0, 1000	1 row(s) returned
8	21:42:27	select * from supply_chain_data where Defect_rates > (select avg(defect_rates) from supply_chain_data)	46 row(s) returned

Problem statement 10

- Get products ranked 6th to 10th by price

The screenshot shows the MySQL Workbench interface with several tabs open:

- Schemas**: Shows the database structure with the **supply_chain** schema selected.
- SQL ASSIGNMENT 2**, **SQL File 3***, **SUPPLY CHAIN DATASET PR...**, **SQL HOMEWORK**, **SUPPLY_CHAIN_DATA PROJE...** (active tab), **Question paper solving(SQL) 1***, and **question paper solving**.
- Result Grid**: Displays the results of the following query:

```
143 -- how customers ranked 11 to 20 by location.
144 • SELECT supplier_Name, Location
145 FROM supply_chain_data
146 ORDER BY Location
147 LIMIT 10 OFFSET 10;
148
149 -- Get products ranked 6th to 10th by price
150 • SELECT "SKU", "Product_type", "Price"
151 FROM supply_chain_data
152 ORDER BY "Price" DESC
153 LIMIT 5 OFFSET 5;
```

The Result Grid shows the following data:

SKU	Product_type	Price
SKU	Product_type	Price

Action Output pane shows the following log:

#	Time	Action	Message	Duration / Fetch
9	00:27:33	SELECT ProductID, REPLACE(ProductType, 'Cosmetics', 'Beauty Products') AS UpdatedProductType FR...	Error Code: 1054. Unknown column 'ProductID' in field list	0.000 sec
10	00:28:02	SELECT "SKU", REPLACE("Inspection results", "Pending", "In-Process") AS UpdatedInspection FROM sup...	100 row(s) returned	23 August 2025
11	00:35:57	SELECT "SKU", "Product_type", "Price" FROM supply_chain_data ORDER BY "Price" DESC LIMIT 5 OFFSE...	5 row(s) returned	Sat 00:36 (Local t)

Problem statement 11

- Find the product with the shortest Manufacturing lead time for each Product type.

The screenshot shows a MySQL Workbench interface with the following details:

- Database:** MySQL80
- Schemas:** A tree view showing various schemas like cab, cnst, dr310, emp_db, healthcare, pizza_sales_analysis, pqrs, sakila, and supply_chain.
- Current Schema:** supply_chain
- Query Editor:** Contains the following SQL code:

```
71
72    -- Find the product with the shortest Manufacturing lead
73    -- time for each Product type.
74
75 • select supply.product_type,supply.sku,supply.manufacturing_lead_time
76   from supply_chain_data supply
77   join (select Product_type,min(Manufacturing_lead_time) as min_lead
78        from supply_chain_data
79       group by Product_type)
80      m on supply.product_type = m.Product_type
81      and supply.Manufacturing_lead_time = m.min_lead;
```
- Result Grid:** Displays the results of the query in a table:

product_type	sku	manufacturing_lead_time
cosmetics	SKU7	1
skincare	SKU40	1
haircare	SKU81	1
- Action Output:** Shows the history of actions taken in the session:

#	Time	Action	Message
8	21:42:27	select * from supply_chain_data where Defect_rates > (select avg(defect_rates) from supply_chain_data) ...	46 row(s) returned
9	21:45:33	select Supplier_name,sum(manufacturing_costs) from supply_chain_data group by Supplier_name LIMIT 0...	5 row(s) returned
10	21:47:46	select supply.product_type,supply.sku,supply.manufacturing_lead_time from supply_chain_data supply join ...	3 row(s) returned

Problem statement 12

- calculate total revenue per location and order it from highest to lowest

The screenshot shows the MySQL Workbench interface with a query editor window. The code is as follows:

```
80 m on supply.product_type = m.Product_type
81 and supply.Manufacturing_lead_time = m.min_lead;
82
83
84 -- calculate total revenue per location and order it
85 -- from highest to lowest
86
87 • select location, sum(revenue_generated) as "total_revenue"
88 from supply_chain_data
89 group by location
90 order by total_revenue desc;
```

The Result Grid shows the output of the query:

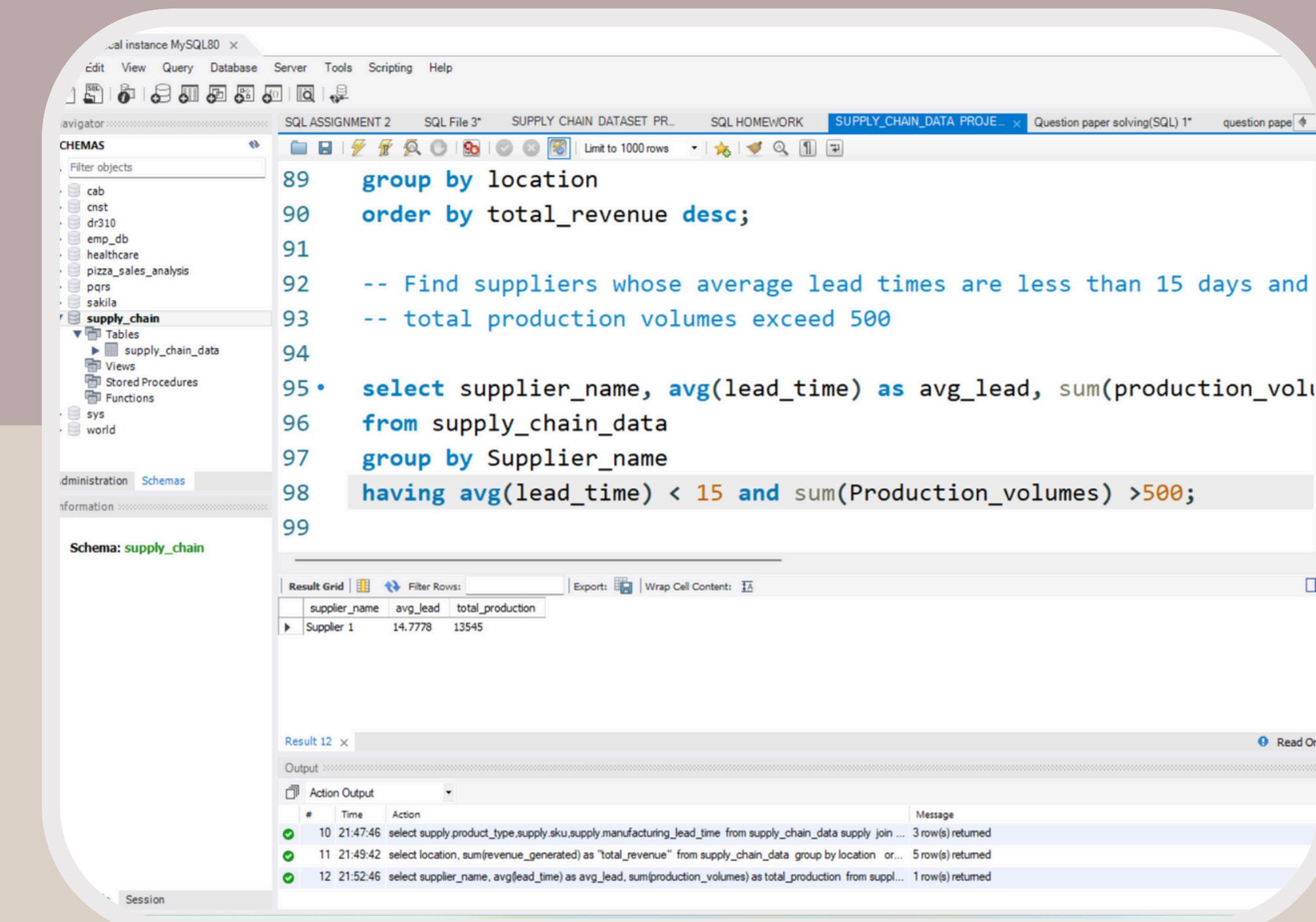
location	total_revenue
Mumbai	137755.02688
Kolkata	137077.55100500002
Chennai	119142.81575000001
Bangalore	102601.72388000002
Delhi	81027.70122500003

The Session tab at the bottom shows the execution history:

#	Time	Action	Message
9	21:45:33	select Supplier_name,sum(manufacturing_costs) from supply_chain_data group by Supplier_name LIMIT 0...	5 row(s) returned
10	21:47:46	select supply.product_type.supply.sku.supply.manufacturing_lead_time from supply_chain_data supply join ...	3 row(s) returned
11	21:49:42	select location, sum(revenue_generated) as "total_revenue" from supply_chain_data group by location or...	5 row(s) returned

Problem statement 13

- Find suppliers whose average lead times are less than 15 days and total production volumes exceed 500



The screenshot shows the MySQL Workbench interface. The left pane displays the 'Schemas' tree, with 'supply_chain' selected. The main pane contains a SQL query:

```
89     group by location
90     order by total_revenue desc;
91
92     -- Find suppliers whose average lead times are less than 15 days and
93     -- total production volumes exceed 500
94
95 • select supplier_name, avg(lead_time) as avg_lead, sum(production_volum
96   from supply_chain_data
97   group by Supplier_name
98   having avg(lead_time) < 15 and sum(Production_volumes) >500;
99
```

The results grid below shows one row of data:

supplier_name	avg_lead	total_production
Supplier 1	14.7778	13545

The 'Output' pane at the bottom shows the execution log:

#	Time	Action	Message
10	21:47:46	select supply.product_type.supply.sku.supply.manufacturing_lead_time from supply_chain_data supply join ...	3 row(s) returned
11	21:49:42	select location, sum(revenue_generated) as "total_revenue" from supply_chain_data group by location or...	5 row(s) returned
12	21:52:46	select supplier_name, avg(lead_time) as avg_lead, sum(production_volumes) as total_production from suppl...	1 row(s) returned

Problem statement 14

- Get the top 3 Transportation modes by total Costs spent.

The screenshot shows the MySQL Workbench interface with a query editor window. The query is:

```
97    group by Supplier_name
98    having avg(lead_time) < 15 and sum(Production_volumes) >500;
99
00    -- Get the top 3 Transportation modes by total Costs spent.
01 • select transportation_modes,sum(costs) as total_cost
02   from supply_chain_data
03   group by Transportation_modes
04   order by total_cost desc
05   limit 3;
06
```

The result grid shows the following data:

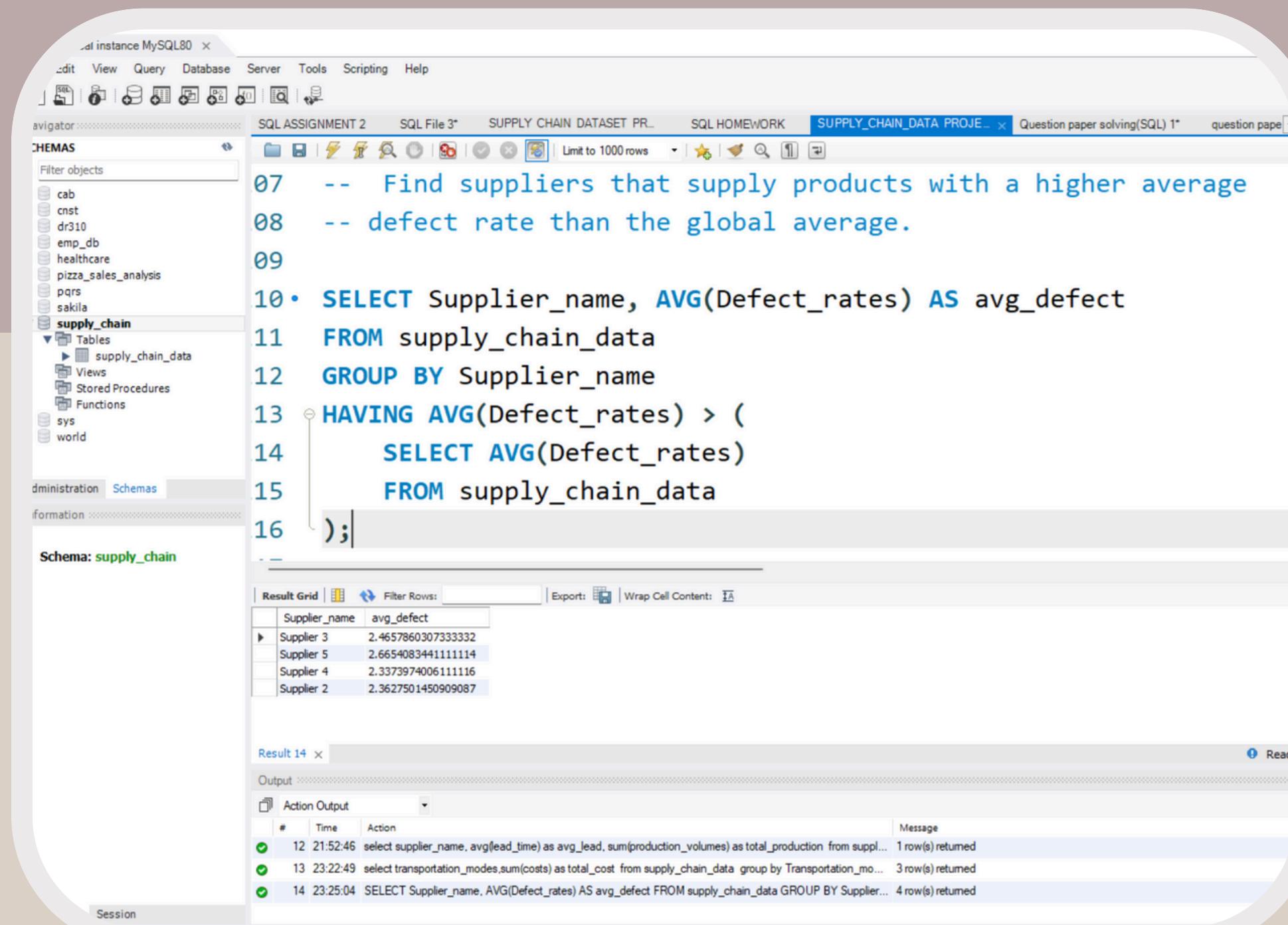
transportation_modes	total_cost
Road	16048.193638899997
Rail	15168.931558400001
Air	14604.5274976

The session output shows the following log entries:

#	Time	Action	Message
11	21:49:42	select location, sum(revenue_generated) as "total_revenue" from supply_chain_data group by location or...	5 row(s) returned
12	21:52:46	select supplier_name, avg(lead_time) as avg_lead, sum(production_volumes) as total_production from suppl...	1 row(s) returned
13	23:22:49	select transportation_modes,sum(costs) as total_cost from supply_chain_data group by Transportation_mo...	3 row(s) returned

Problem statement 15

- Find suppliers that supply products with a higher average defect rate than the global average.



The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The "supply_chain" schema is selected.
- Query Editor:** Contains the following SQL code:

```
07  -- Find suppliers that supply products with a higher average
08  -- defect rate than the global average.
09
10 • SELECT Supplier_name, AVG(Defect_rates) AS avg_defect
11 FROM supply_chain_data
12 GROUP BY Supplier_name
13 HAVING AVG(Defect_rates) > (
14     SELECT AVG(Defect_rates)
15     FROM supply_chain_data
16 );
```
- Result Grid:** Displays the results of the query:

Supplier_name	avg_defect
Supplier 3	2.465786030733332
Supplier 5	2.6654083441111114
Supplier 4	2.3373974006111116
Supplier 2	2.3627501450909087
- Action Output:** Shows the log of actions taken:

#	Time	Action	Message
12	21:52:46	select supplier_name, avg(lead_time) as avg_lead, sum(production_volumes) as total_production from suppl...	1 row(s) returned
13	23:22:49	select transportation_modes.sum(costs) as total_cost from supply_chain_data group by Transportation_mo...	3 row(s) returned
14	23:25:04	SELECT Supplier_name, AVG(Defect_rates) AS avg_defect FROM supply_chain_data GROUP BY Supplier...	4 row(s) returned

Future Scope

- Integration with Real-Time Data: Build live dashboards for continuous supply chain tracking.
- Predictive Analytics: Use machine learning to forecast demand, lead times, and defect rates.
- Optimization Techniques: Minimize logistics costs and improve transportation mode selection.
- Sustainability Tracking: Monitor eco-friendly practices, carbon footprint, and supplier compliance.
- Global Expansion: Extend analysis to international supply chains with diverse markets and regulations.

Conclusion

- SQL analysis provided deep insights into product performance, customer demographics, supplier efficiency, and logistics.
- Identified top-performing products and locations contributing maximum revenue.
- Highlighted operational challenges such as defect rates, lead times, and high transportation costs.
- Demonstrated the value of data-driven decision-making in supply chain management.
- Strengthened understanding of the end-to-end supply chain for skincare, haircare, and cosmetics.

thank you!