

Online Shopping Platform



-By Akshay Gujare
77418analyst@gmail.com

ONLINE SHOPPING PLATFORM

SQL Assignment

We are analyzing a small online shopping platform where customers register and place orders. The company wants to track customer details and their purchases for reporting and analysis.

- Database Name: **RetailDB_1**
- Purpose: To understand who the customers are, when they joined, and what orders they placed.
- At this stage, we are only storing customer information and their orders.

Tables in RetailDB:

1. **Customers** – stores details of each registered customer.

(**customer_id**, name, email, city, signup_date)

2. **Products** - holds all the products details

(**product_id**, product_name, category, price)

3. **Orders** – keeps records of every order placed, linked to the customer.

(**order_id**, customer_id, product_id, order_date, quantity, total_amount, payment_mode)

Orders

order_id	customer_id	product_id	order_date	quantity	total_amount	payment_mode
1	1	1	2024-07-10	1	79999.00	UPI
2	2	2	2024-07-15	1	74999.00	Credit Card
3	3	3	2024-07-20	1	4999.00	Net Banking
4	4	6	2024-07-22	2	5998.00	Debit Card
5	5	5	2024-08-01	1	1999.00	UPI
6	6	7	2024-08-03	1	6499.00	Credit Card
7	8	10	2024-08-05	1	59999.00	Debit Card
8	9	8	2024-08-10	3	1497.00	UPI

Customers

customer_id	name	email	city	signup_date
1	Rohit Kumar	rohit.kumar@gmail.com	Delhi	2024-01-15
2	Sneha Sharma	sneha.sharma@yahoo.com	Mumbai	2024-02-20
3	Amit Patel	amit.patel@gmail.com	Ahmedabad	2024-03-05
4	Priya Reddy	priya.reddy@gmail.com	Hyderabad	2024-03-22
5	Karan Singh	karan.singh@outlook.com	Chennai	2024-04-10
6	Neha Verma	neha.verma@gmail.com	Pune	2024-05-08
7	Arjun Mehta	arjun.mehta@gmail.com	Jaipur	2024-05-15
8	Ananya Iyer	ananya.iyer@gmail.com	Bengaluru	2024-06-02
9	Vikram Das	vikram.das@gmail.com	Kolkata	2024-06-20
10	Meera Nair	meera.nair@gmail.com	Kochi	2024-07-01

Products

product_id	product_name	category	price
1	iPhone 15	Electronics	79999.00
2	Samsung Galaxy S24	Electronics	74999.00
3	Noise Smartwatch	Wearables	4999.00
4	Boat Earbuds	Wearables	2499.00
5	Kurta Set	Fashion	1999.00
6	Running Shoes	Fashion	2999.00
7	Prestige Mixer Grinder	Home Appliances	6499.00
8	Tata Tea 1kg Pack	Groceries	499.00
9	Amul Butter 500g	Groceries	285.00
10	Sony Bravia 55" TV	Electronics	59999.00

Task: Solve the below mentioned questions by writing SQL queries

1. Fetch all customers from the database.
2. Show only the customer names and their cities.
3. Find customers who live in Mumbai.
4. Get all orders placed after 1st August 2024.
5. List all products priced greater than ₹5000.
6. Count how many customers exist in the system.
7. Update a customer's city (e.g., change Rohit Kumar's city to Hyderabad).
8. Delete an order (e.g., remove order with ID = 5).
9. Display product names with their original price and price increased by 10%.
10. Show only the unique cities where customers live.
11. Get the first 3 customers who signed up.
12. Skip the first 2 customers and fetch the next 3 customers.
13. Find products with prices between ₹2000 and ₹6000.
14. Find customers who are from Mumbai OR Chennai.
15. Find customers who are NOT from Delhi.
16. Find orders that are NOT paid by UPI.
17. Get the average order amount across all orders.
18. Show the highest order amount.
19. Show the lowest product price.
20. Find the total money spent across all orders

Table creation and Import of data from csv file

Step 1: Database creation

```
1 • Create database RetailDB_1;  
2 • use RetailDB_1;
```



Step 3: Import data from CSV file

```
SCHEMAS  
Filter objects  
retaildb_1  
Tables  
customers  
Select Rows - Limit 500  
Table Inspector  
Copy to Clipboard  
Table Data Export Wizard  
Table Data Import Wizard  
Send to SQL Editor  
Create Table...  
Create Table Like...  
Alter Table...  
Table Maintenance...  
Drop Table...  
Truncate Table...  
Search Table Data...  
Refresh All  
customer_id int  
name varchar(100)  
email varchar(150)  
city varchar(50)  
signup_date date  
1 • Create database RetailDB_1;  
2 • use RetailDB_1;  
3 • CREATE TABLE Customers (
```

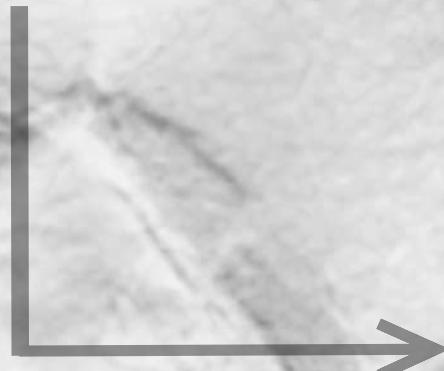
Step 2: Table creation (Customer)

```
3 • CREATE TABLE Customers (  
4     customer_id INT AUTO_INCREMENT PRIMARY KEY,  
5     name VARCHAR(100),  
6     email VARCHAR(150) UNIQUE,  
7     city VARCHAR(50),  
8     signup_date DATE  
9 );
```



Step 4: Display Imported data

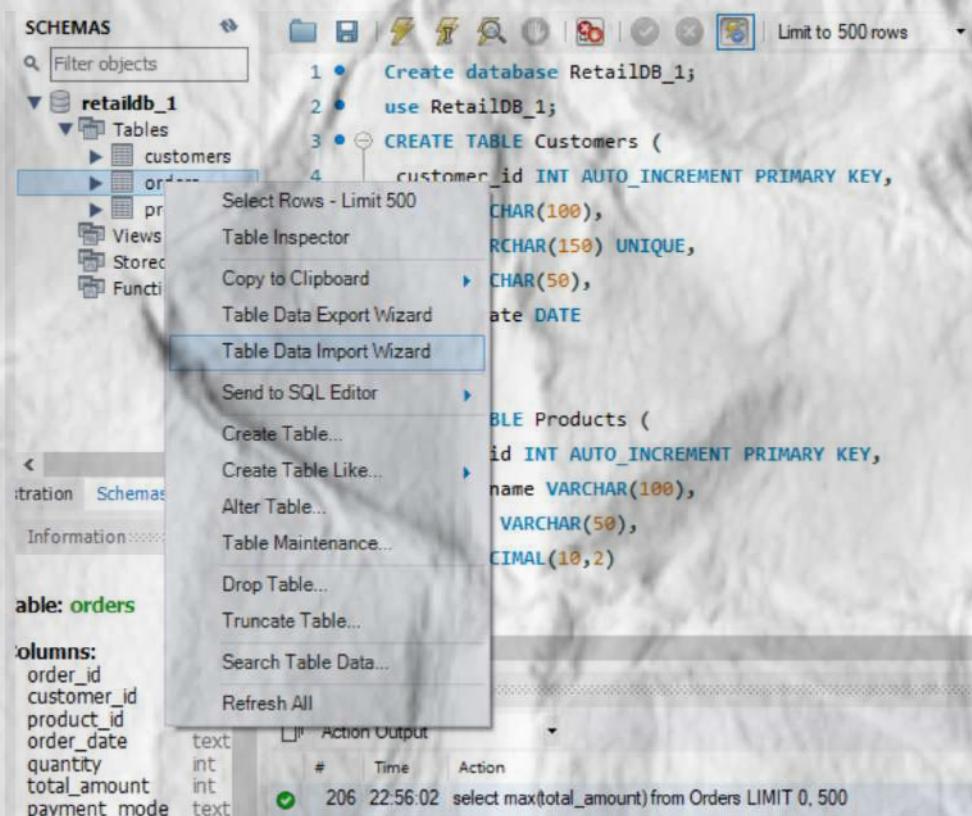
```
29 • SELECT * FROM Customers;  
30  
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |  
customer_id name email city signup_date  
1 Rohit Kumar rohit.kumar@gmail.com Delhi 2023-01-12  
2 Sneha Sharma sneha.sharma@yahoo.com Mumbai 2023-02-05  
3 Amit Patel amit.patel@gmail.com Ahmedabad 2023-02-18  
4 Priya Reddy priya.reddy@gmail.com Hyderabad 2023-03-02  
5 Karan Singh karan.singh@outlook.com Chennai 2023-03-15  
6 Neha Verma neha.verma@gmail.com Pune 2023-04-01  
7 Arjun Mehta arjun.mehta@gmail.com Bengaluru 2023-04-20  
8 Ritika Gupta ritika.gupta@yahoo.com Kolkata 2023-05-12  
9 Vikram Joshi vikram.joshi@gmail.com Lucknow 2023-05-25  
10 Ananya Das ananya.das@gmail.com Bhubaneswar 2023-06-08  
11 Suresh Iyer suresh.iyer@gmail.com Chennai 2023-06-20  
12 Megha Kapoor megha.kapoor@yahoo.com Jaipur 2023-07-03  
13 Ravi Shankar ravi.shankar@gmail.com Delhi 2023-07-15  
14 Tanya Mishra tanya.mishra@gmail.com Noida 2023-08-01  
15 Aditya Jain aditya.jain@gmail.com Indore 2023-08-14  
NULL NULL NULL NULL NULL
```



Step 5: Table creation (Orders)

```
18 • CREATE TABLE Orders (
19     order_id INT AUTO_INCREMENT PRIMARY KEY,
20     customer_id INT,
21     product_id INT,
22     order_date DATE,
23     quantity INT,
24     total_amount DECIMAL(10,2),
25     payment_mode VARCHAR(50),
26     FOREIGN KEY (customer_id) REFERENCES Customers(customer_id),
27     FOREIGN KEY (product_id) REFERENCES Products(product_id)
28 );
```

Step 6: Import data from CSV file



SCHEMAS

retaildb_1

Tables

customers

orders

Views

Stored Procedures

Functions

CREATE TABLE Customers (

customer_id INT AUTO_INCREMENT PRIMARY KEY,

CHAR(100),

RCHAR(150) UNIQUE,

CHAR(50),

date DATE

Select Rows - Limit 500

Table Inspector

Copy to Clipboard

Table Data Export Wizard

Table Data Import Wizard

Send to SQL Editor

Create Table...

Create Table Like...

Alter Table...

Table Maintenance...

Drop Table...

Truncate Table...

Search Table Data...

Refresh All

able: orders

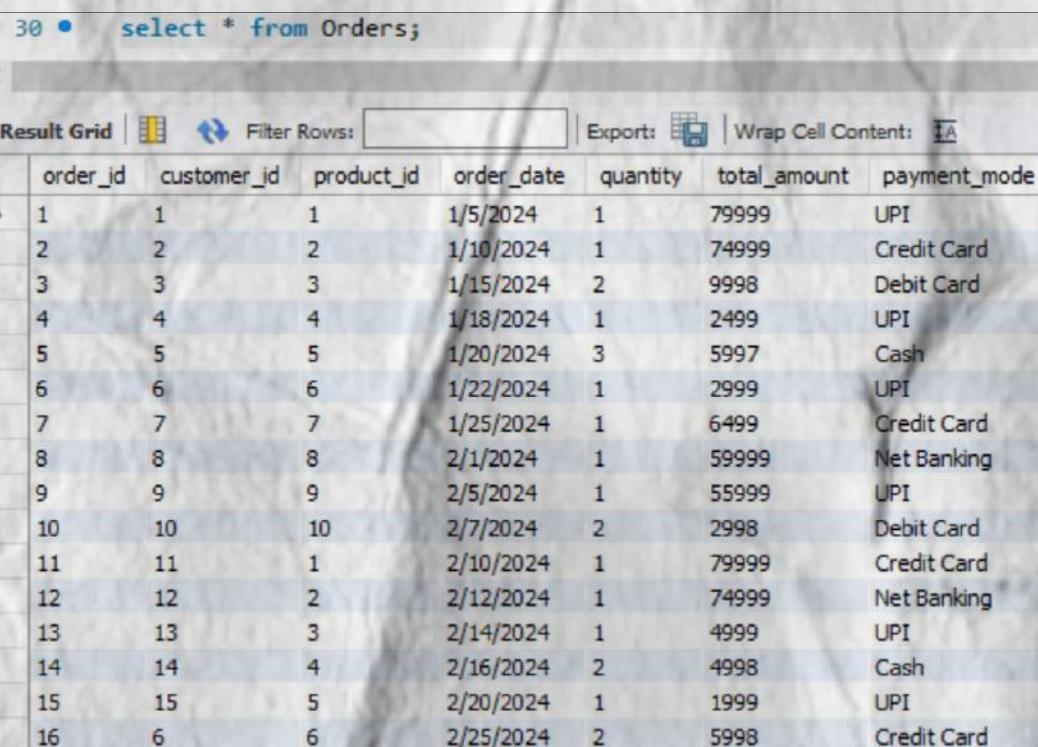
columns:

order_id	customer_id	product_id	order_date	quantity	total_amount	payment_mode
text	int	int	text	#	Time	Action

206 22:56:02 select max(total_amount) from Orders LIMIT 0, 500

Step 7: Display Imported data

```
30 • select * from Orders;
```

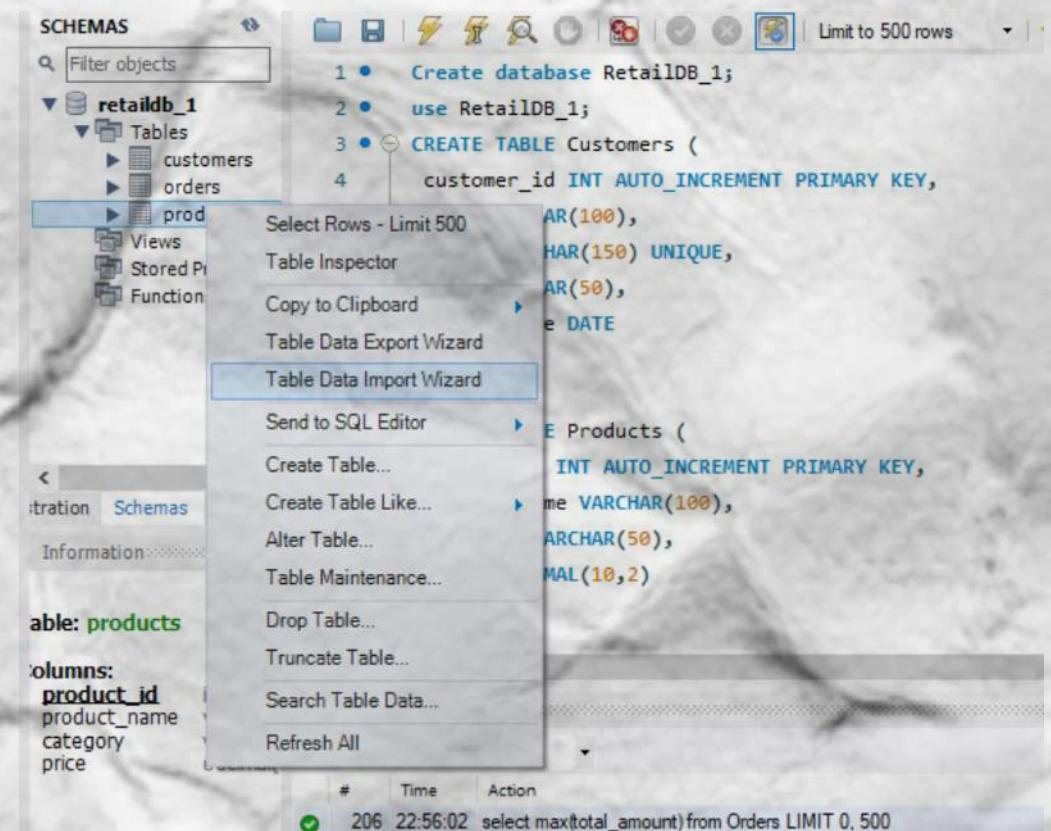


	order_id	customer_id	product_id	order_date	quantity	total_amount	payment_mode
1	1	1	1	1/5/2024	1	79999	UPI
2	2	2	2	1/10/2024	1	74999	Credit Card
3	3	3	3	1/15/2024	2	9998	Debit Card
4	4	4	4	1/18/2024	1	2499	UPI
5	5	5	5	1/20/2024	3	5997	Cash
6	6	6	6	1/22/2024	1	2999	UPI
7	7	7	7	1/25/2024	1	6499	Credit Card
8	8	8	8	2/1/2024	1	59999	Net Banking
9	9	9	9	2/5/2024	1	55999	UPI
10	10	10	10	2/7/2024	2	2998	Debit Card
11	11	1	2	2/10/2024	1	79999	Credit Card
12	12	2	2	2/12/2024	1	74999	Net Banking
13	13	3	3	2/14/2024	1	4999	UPI
14	14	4	4	2/16/2024	2	4998	Cash
15	15	5	5	2/20/2024	1	1999	UPI
16	6	6	6	2/25/2024	2	5998	Credit Card

Step 8: Table creation (Products)

```
11 • CREATE TABLE Products (
12     product_id INT AUTO_INCREMENT PRIMARY KEY,
13     product_name VARCHAR(100),
14     category VARCHAR(50),
15     price DECIMAL(10,2)
16 );
```

Step 9: Import data from CSV file



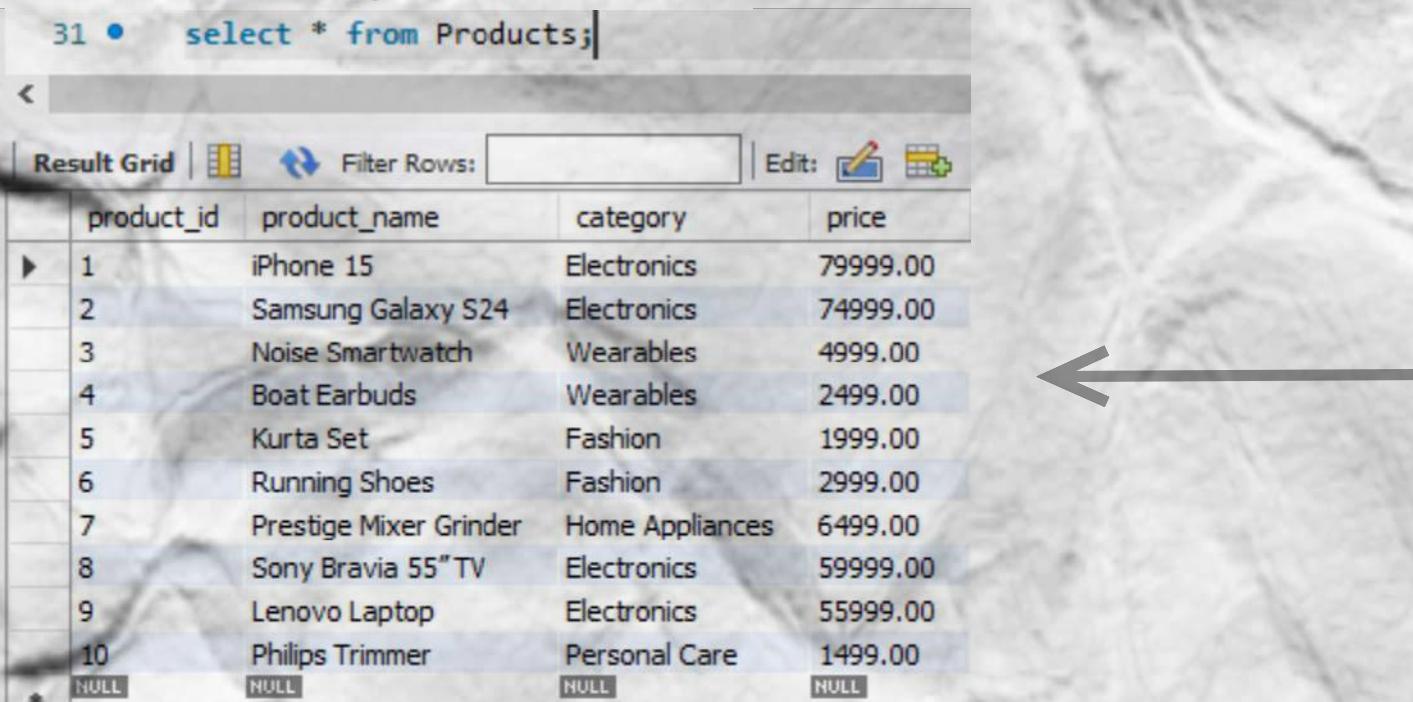
The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' tree shows the 'retaildb_1' schema with tables 'customers', 'orders', and 'prod'. The 'prod' table is selected. On the right, the SQL editor contains the creation of the 'Products' table:

```
1 • Create database RetailDB_1;
2 • use RetailDB_1;
3 • CREATE TABLE Customers (
4     customer_id INT AUTO_INCREMENT PRIMARY KEY,
5     customer_name VARCHAR(100),
6     customer_email UNIQUE,
7     customer_address VARCHAR(50),
8     customer_date DATE
9 );
10 • CREATE TABLE Products (
11     product_id INT AUTO_INCREMENT PRIMARY KEY,
12     product_name VARCHAR(100),
13     category VARCHAR(50),
14     price DECIMAL(10,2)
15 );
```

The 'Table Data Import Wizard' is open for the 'products' table. It lists columns: 'product_id', 'product_name', 'category', and 'price'. The 'Action' bar at the bottom shows a successful query: # 206 22:56:02 select max(total_amount)from Orders LIMIT 0,500.

Step 10: Display Imported data

```
31 • select * from Products;
```



The screenshot shows the MySQL Workbench interface with the results of the 'select * from Products' query. The results are displayed in a 'Result Grid' table:

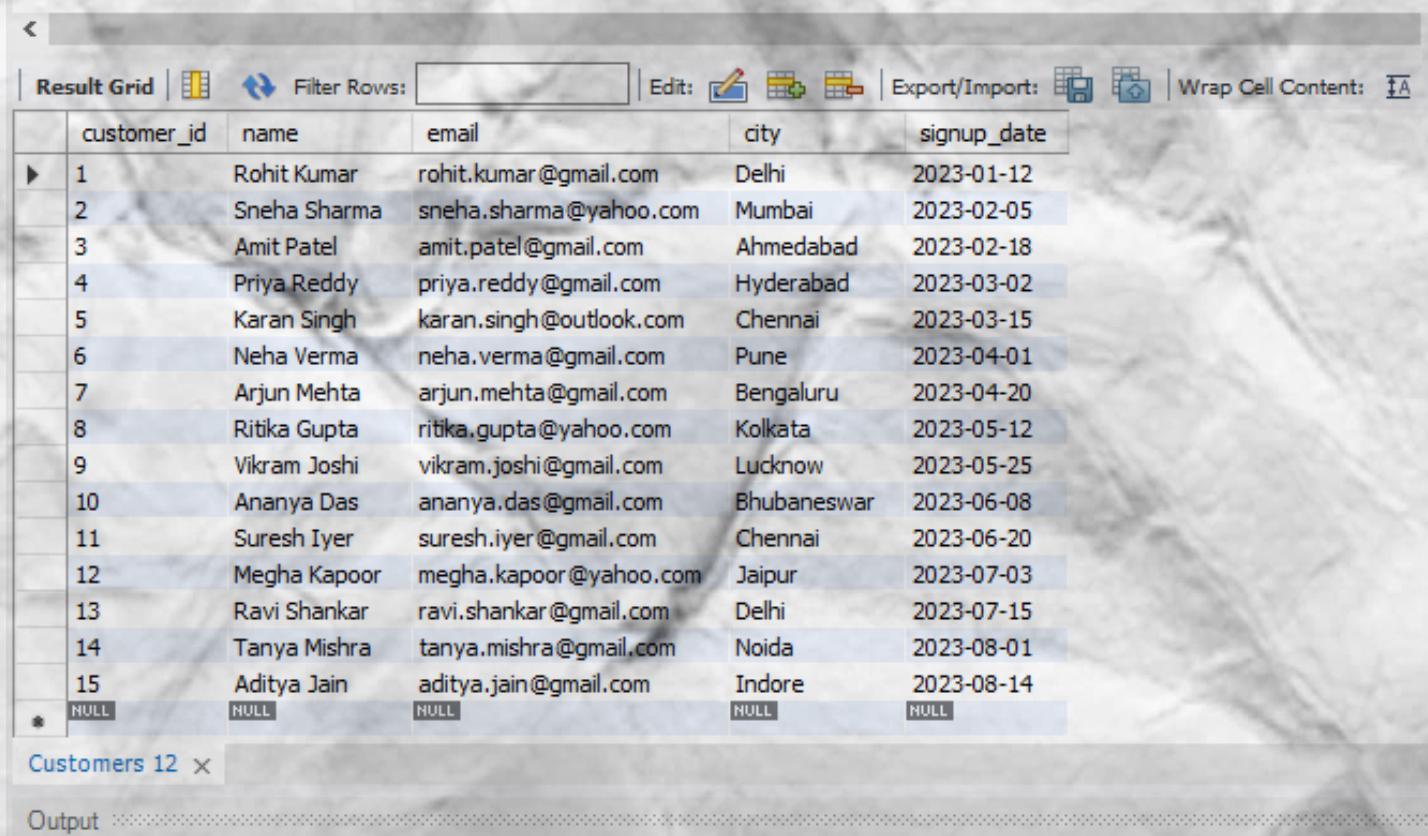
	product_id	product_name	category	price
1	1	iPhone 15	Electronics	79999.00
2	2	Samsung Galaxy S24	Electronics	74999.00
3	3	Noise Smartwatch	Wearables	4999.00
4	4	Boat Earbuds	Wearables	2499.00
5	5	Kurta Set	Fashion	1999.00
6	6	Running Shoes	Fashion	2999.00
7	7	Prestige Mixer Grinder	Home Appliances	6499.00
8	8	Sony Bravia 55"TV	Electronics	59999.00
9	9	Lenovo Laptop	Electronics	55999.00
10	10	Philips Trimmer	Personal Care	1499.00
*		NULL	NULL	NULL

Finally all three tables of Customers ,Orders & Products with data is ready to perform given operations or task.

Solved problem statements

1. Fetch all customers from the database.

```
33      -- 1 Fetch all customers from the database.  
34 •  SELECT * FROM Customers;  
35
```



The screenshot shows a database query results window. At the top, there is a code editor with the following SQL query:

```
33      -- 1 Fetch all customers from the database.  
34 •  SELECT * FROM Customers;  
35
```

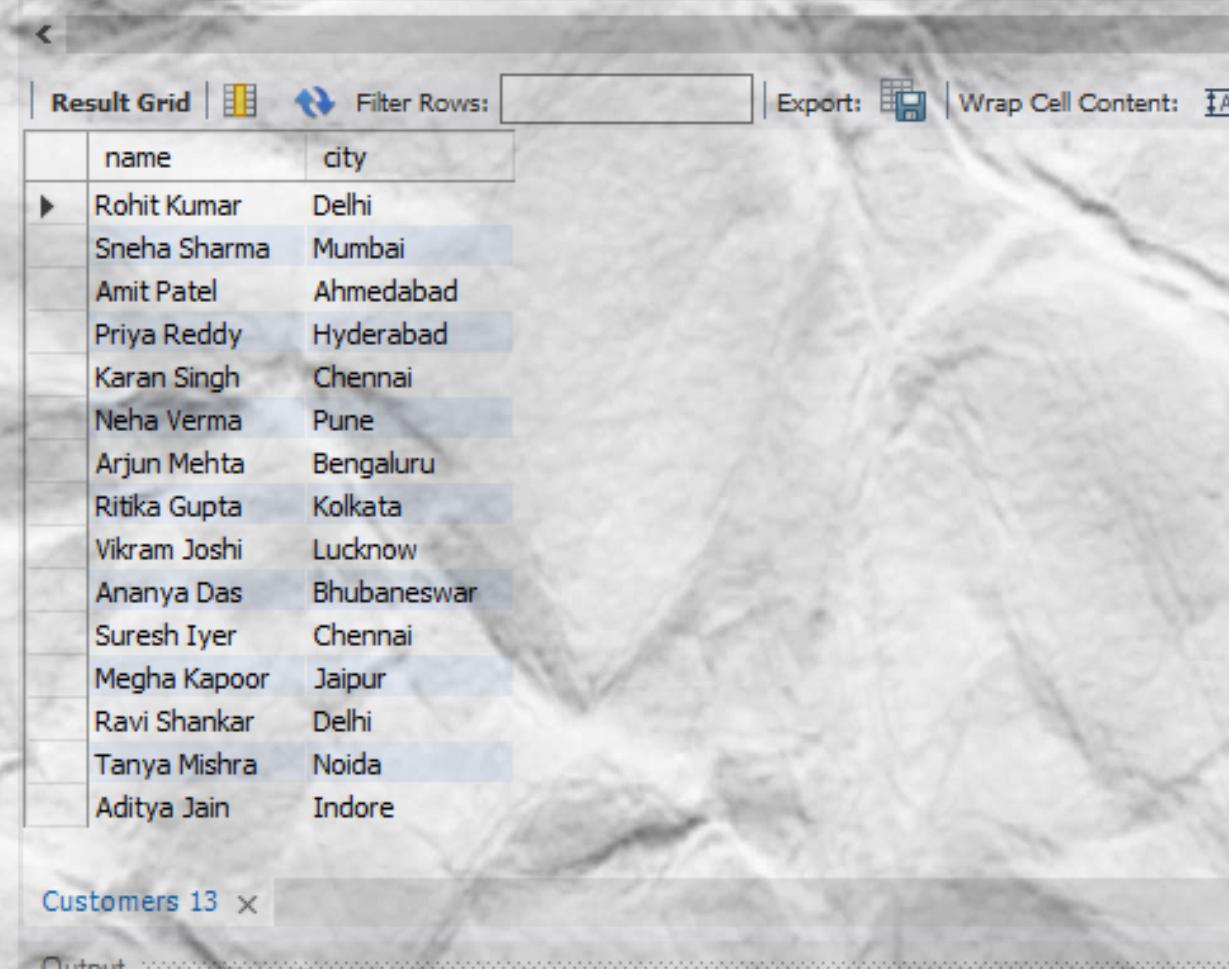
Below the code editor is a results grid. The grid has the following structure:

	customer_id	name	email	city	signup_date
▶	1	Rohit Kumar	rohit.kumar@gmail.com	Delhi	2023-01-12
	2	Sneha Sharma	sneha.sharma@yahoo.com	Mumbai	2023-02-05
	3	Amit Patel	amit.patel@gmail.com	Ahmedabad	2023-02-18
	4	Priya Reddy	priya.reddy@gmail.com	Hyderabad	2023-03-02
	5	Karan Singh	karan.singh@outlook.com	Chennai	2023-03-15
	6	Neha Verma	neha.verma@gmail.com	Pune	2023-04-01
	7	Arjun Mehta	arjun.mehta@gmail.com	Bengaluru	2023-04-20
	8	Ritika Gupta	ritika.gupta@yahoo.com	Kolkata	2023-05-12
	9	Vikram Joshi	vikram.joshi@gmail.com	Lucknow	2023-05-25
	10	Ananya Das	ananya.das@gmail.com	Bhubaneswar	2023-06-08
	11	Suresh Iyer	suresh.iyer@gmail.com	Chennai	2023-06-20
	12	Megha Kapoor	megha.kapoor@yahoo.com	Jaipur	2023-07-03
	13	Ravi Shankar	ravi.shankar@gmail.com	Delhi	2023-07-15
	14	Tanya Mishra	tanya.mishra@gmail.com	Noida	2023-08-01
	15	Aditya Jain	aditya.jain@gmail.com	Indore	2023-08-14
*	NULL	NULL	NULL	NULL	NULL

The results grid has a header bar with buttons for 'Result Grid', 'Filter Rows', 'Edit', 'Export/Import', and 'Wrap Cell Content'. Below the grid, there is a tab labeled 'Customers 12' and an 'Output' section.

2. Show only the customer names and their cities.

```
36      -- 2. Show only the customer names and their cities.  
37 •  Select name,city from Customers;
```



The screenshot shows a database query results window. At the top, there is a code editor with the following SQL query:

```
36      -- 2. Show only the customer names and their cities.  
37 •  Select name,city from Customers;
```

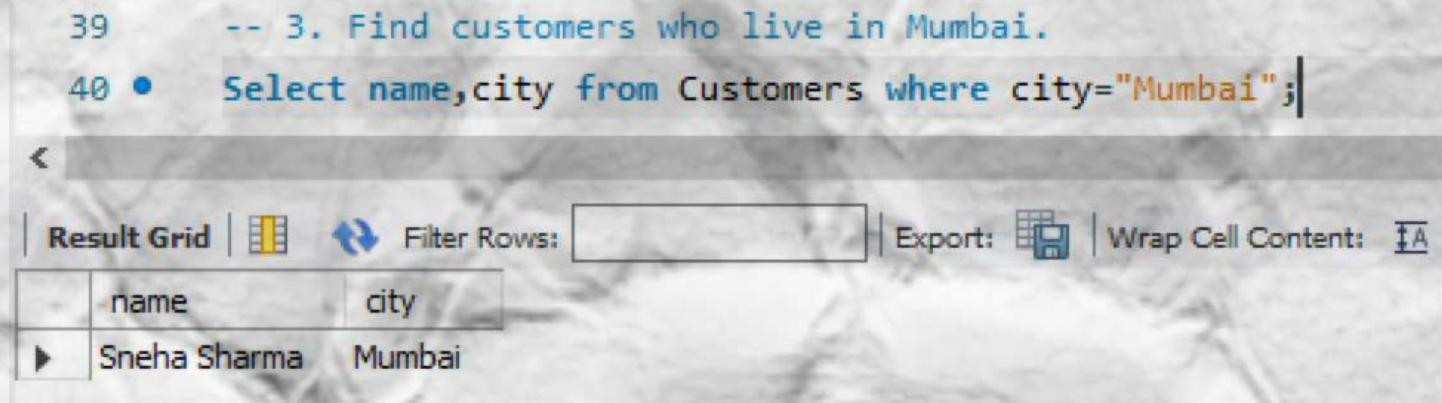
Below the code editor is a results grid. The grid has the following structure:

	name	city
▶	Rohit Kumar	Delhi
	Sneha Sharma	Mumbai
	Amit Patel	Ahmedabad
	Priya Reddy	Hyderabad
	Karan Singh	Chennai
	Neha Verma	Pune
	Arjun Mehta	Bengaluru
	Ritika Gupta	Kolkata
	Vikram Joshi	Lucknow
	Ananya Das	Bhubaneswar
	Suresh Iyer	Chennai
	Megha Kapoor	Jaipur
	Ravi Shankar	Delhi
	Tanya Mishra	Noida
	Aditya Jain	Indore

The results grid has a header bar with buttons for 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'. Below the grid, there is a tab labeled 'Customers 13' and an 'Output' section.

3. Find customers who live in Mumbai.

```
39      -- 3. Find customers who live in Mumbai.  
40 •  Select name,city from Customers where city="Mumbai";
```

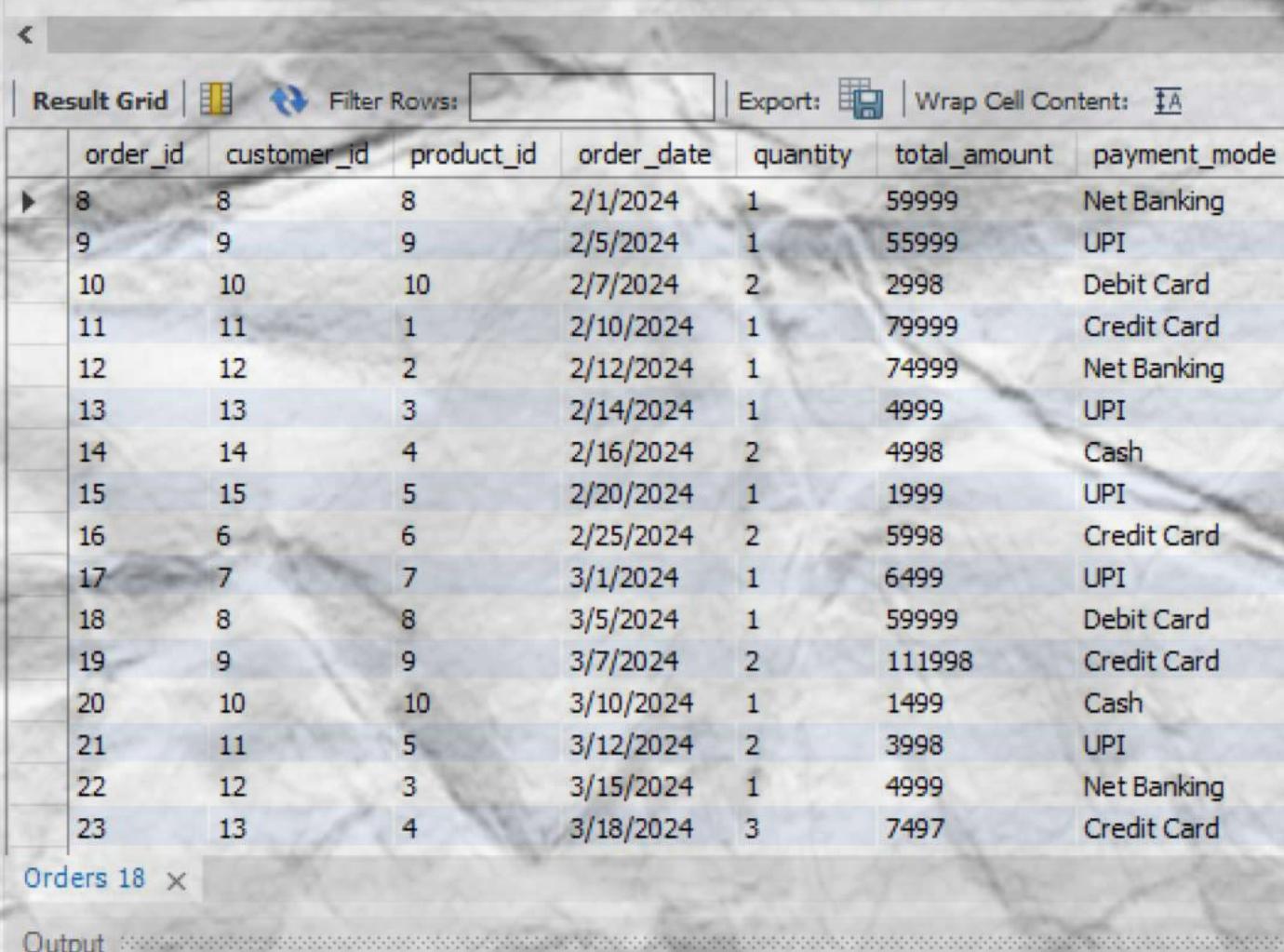


The screenshot shows a MySQL query results window. The query is: "Select name,city from Customers where city='Mumbai';". The result grid displays one row with the columns 'name' and 'city'. The data is: name = 'Sneha Sharma' and city = 'Mumbai'.

	name	city
▶	Sneha Sharma	Mumbai

4. Get all orders placed after 1st August 2024.

```
42      -- 4. Get all orders placed after 1st August 2024.  
43 •  Select * from Orders WHERE order_date>"1/8/2024";
```



The screenshot shows a MySQL query results window. The query is: "Select * from Orders WHERE order_date>'1/8/2024';". The result grid displays 18 rows of order data with columns: order_id, customer_id, product_id, order_date, quantity, total_amount, and payment_mode. The data includes various order details such as order_id 8, customer_id 8, product_id 8, order_date 2/1/2024, quantity 1, total_amount 59999, and payment_mode Net Banking.

	order_id	customer_id	product_id	order_date	quantity	total_amount	payment_mode
▶	8	8	8	2/1/2024	1	59999	Net Banking
	9	9	9	2/5/2024	1	55999	UPI
	10	10	10	2/7/2024	2	2998	Debit Card
	11	11	1	2/10/2024	1	79999	Credit Card
	12	12	2	2/12/2024	1	74999	Net Banking
	13	13	3	2/14/2024	1	4999	UPI
	14	14	4	2/16/2024	2	4998	Cash
	15	15	5	2/20/2024	1	1999	UPI
	16	6	6	2/25/2024	2	5998	Credit Card
	17	7	7	3/1/2024	1	6499	UPI
	18	8	8	3/5/2024	1	59999	Debit Card
	19	9	9	3/7/2024	2	111998	Credit Card
	20	10	10	3/10/2024	1	1499	Cash
	21	11	5	3/12/2024	2	3998	UPI
	22	12	3	3/15/2024	1	4999	Net Banking
	23	13	4	3/18/2024	3	7497	Credit Card

Orders 18 ×

Output

5. List all products priced greater than ₹5000.

```
45      -- 5. List all products priced greater than ₹5000.  
46 •  Select product_name, price from Products where price >5000;
```

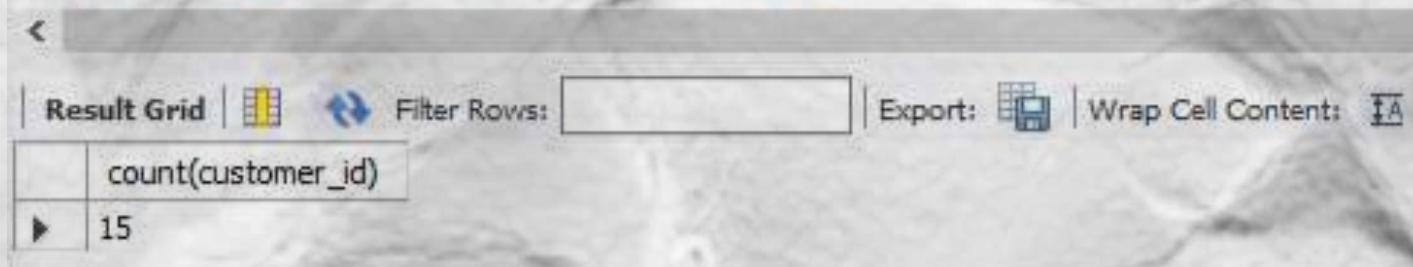


A screenshot of a MySQL query results grid. The grid has a header row with 'product_name' and 'price' columns. Below the header, there are five data rows. The data is as follows:

	product_name	price
▶	iPhone 15	79999.00
	Samsung Galaxy S24	74999.00
	Prestige Mixer Grinder	6499.00
	Sony Bravia 55" TV	59999.00
	Lenovo Laptop	55999.00

6. Count how many customers exist in the system.

```
48      -- 6. Count how many customers exist in the system.  
49 •  Select count(customer_id) from Customers;
```

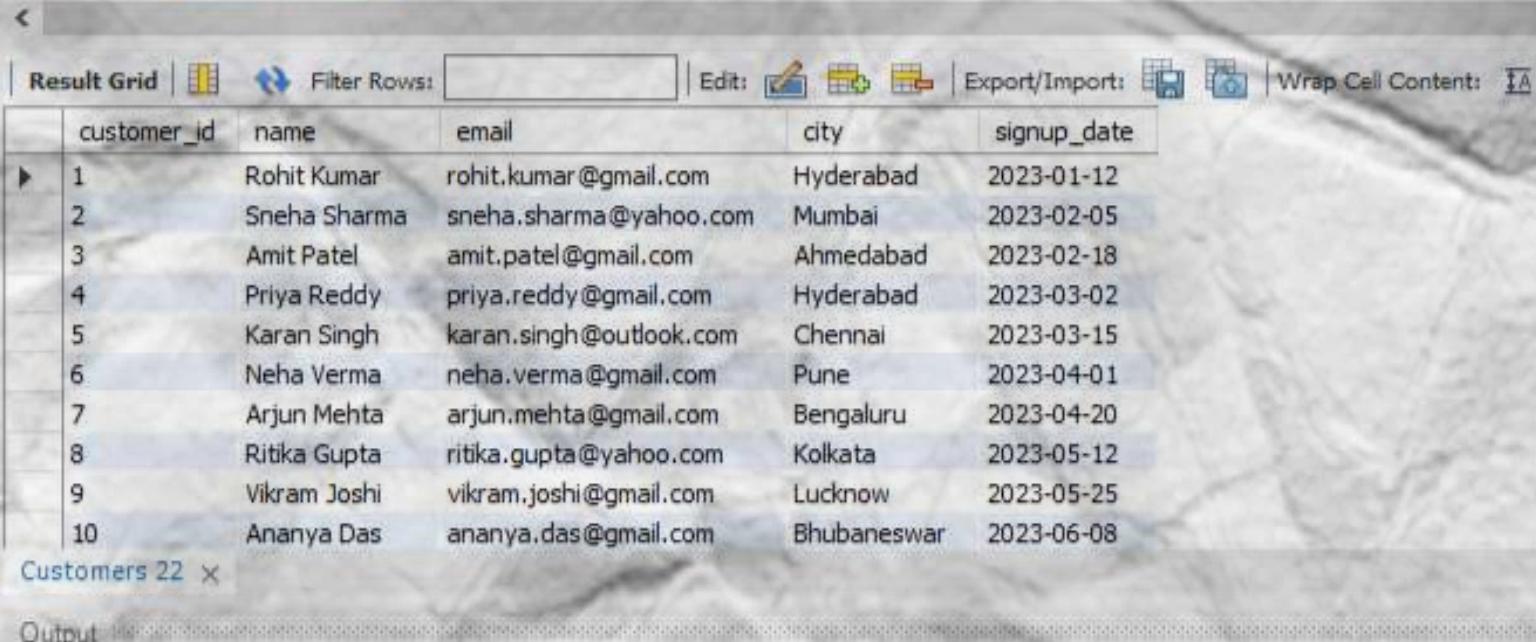


A screenshot of a MySQL query results grid. The grid has a header row with 'count(customer_id)' and a single data row below it. The data is as follows:

	count(customer_id)
▶	15

7. Update a customer's city (e.g., change Rohit Kumar's city to Hyderabad).

```
51      -- 7. Update a customer's city (e.g., change Rohit Kumar's city to Hyderabad).  
52 •  set sql_safe_updates=0;  
53 •  update Customers set city="Hyderabad" where name="Rohit Kumar";  
54 •  select * from Customers;
```



A screenshot of a MySQL query results grid. The grid has a header row with 'customer_id', 'name', 'email', 'city', and 'signup_date' columns. Below the header, there are ten data rows. The data is as follows:

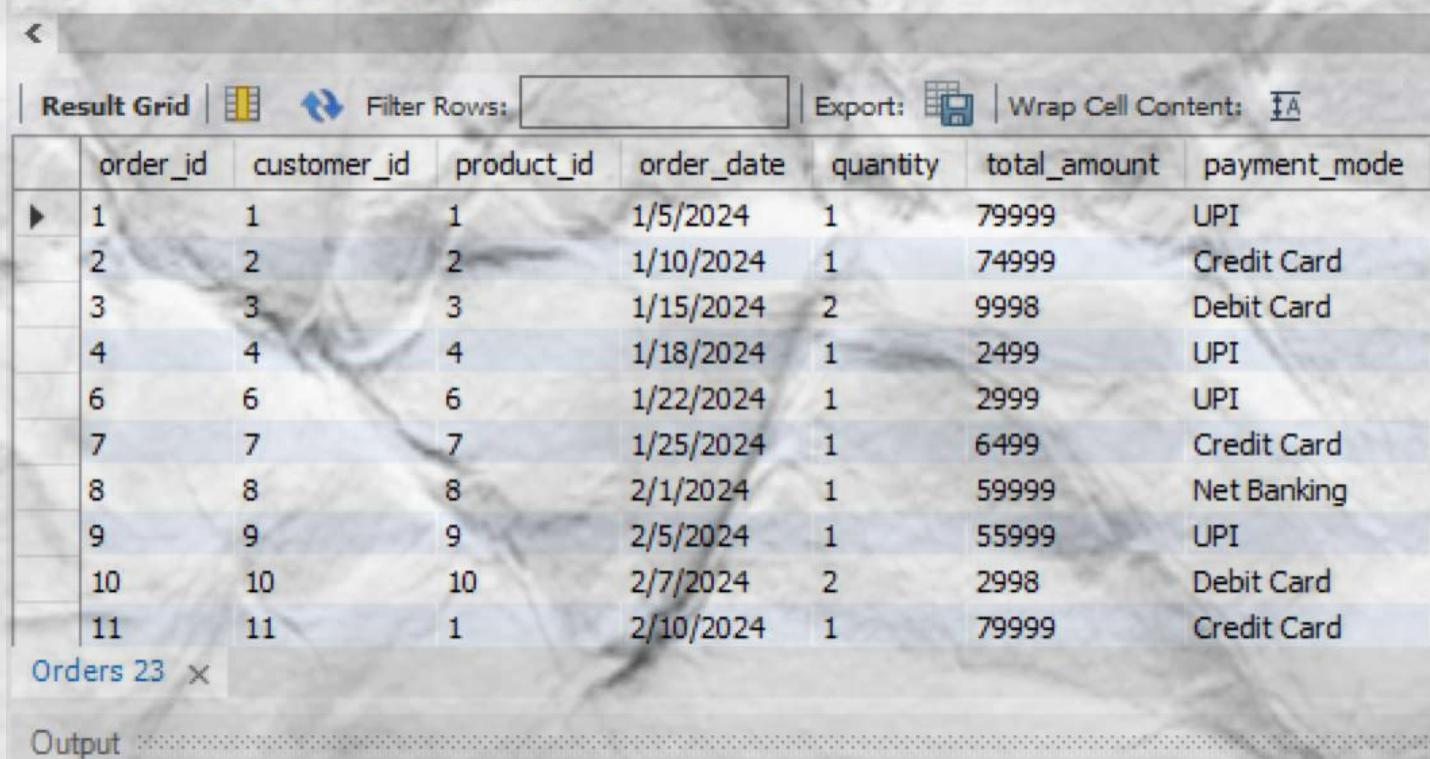
	customer_id	name	email	city	signup_date
▶	1	Rohit Kumar	rohit.kumar@gmail.com	Hyderabad	2023-01-12
	2	Sneha Sharma	sneha.sharma@yahoo.com	Mumbai	2023-02-05
	3	Amit Patel	amit.patel@gmail.com	Ahmedabad	2023-02-18
	4	Priya Reddy	priya.reddy@gmail.com	Hyderabad	2023-03-02
	5	Karan Singh	karan.singh@outlook.com	Chennai	2023-03-15
	6	Neha Verma	neha.verma@gmail.com	Pune	2023-04-01
	7	Arjun Mehta	arjun.mehta@gmail.com	Bengaluru	2023-04-20
	8	Ritika Gupta	ritika.gupta@yahoo.com	Kolkata	2023-05-12
	9	Vikram Joshi	vikram.joshi@gmail.com	Lucknow	2023-05-25
	10	Ananya Das	ananya.das@gmail.com	Bhubaneswar	2023-06-08

Customers 22 ×

Output

8. Delete an order (e.g., remove order with ID = 5).

```
56      -- 8. Delete an order (e.g., remove order with ID = 5).
57 •  delete from Orders where order_id=5;
58 •  select * from Orders;
```

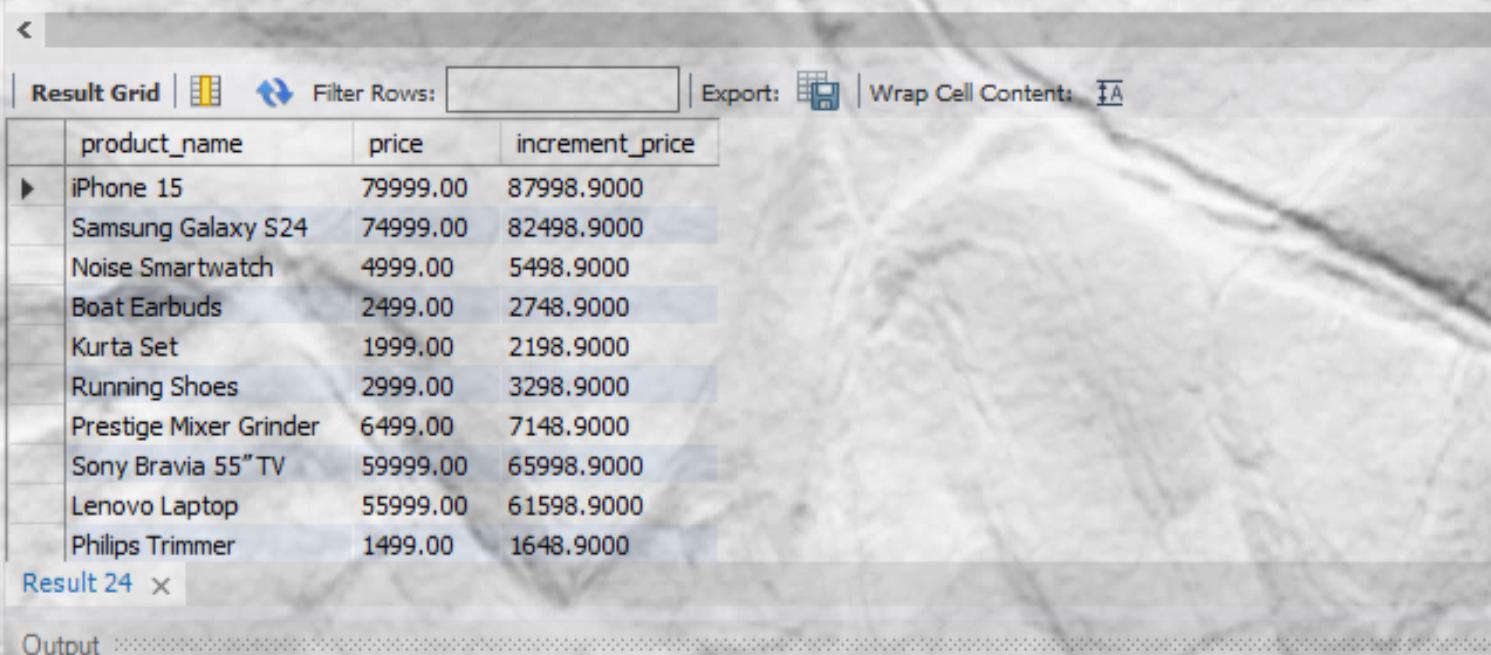


The screenshot shows the MySQL Workbench interface with the 'Orders' table selected. The table has columns: order_id, customer_id, product_id, order_date, quantity, total_amount, and payment_mode. The data consists of 11 rows, each representing an order with unique values for all columns. The table is displayed in a grid format with a light gray background and white text. The 'Result Grid' tab is active, and there are buttons for 'Filter Rows', 'Export', and 'Wrap Cell Content'.

	order_id	customer_id	product_id	order_date	quantity	total_amount	payment_mode
▶	1	1	1	1/5/2024	1	79999	UPI
	2	2	2	1/10/2024	1	74999	Credit Card
	3	3	3	1/15/2024	2	9998	Debit Card
	4	4	4	1/18/2024	1	2499	UPI
	6	6	6	1/22/2024	1	2999	UPI
	7	7	7	1/25/2024	1	6499	Credit Card
	8	8	8	2/1/2024	1	59999	Net Banking
	9	9	9	2/5/2024	1	55999	UPI
	10	10	10	2/7/2024	2	2998	Debit Card
	11	11	1	2/10/2024	1	79999	Credit Card

9. Display product names with their original price and price increased by 10%.

```
60      -- 9. Display product names with their original price and price increased by 10%.
61 •  Select product_name,price,(price*1.10) as increment_price from Products;
62
```

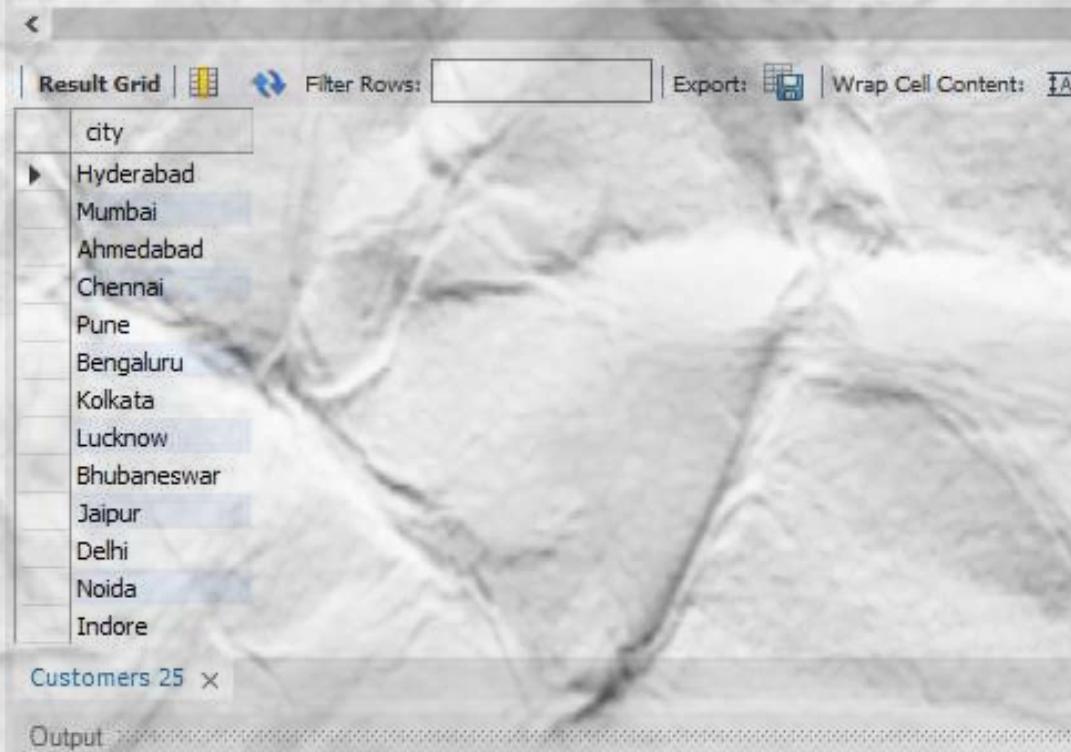


The screenshot shows the MySQL Workbench interface with the 'Products' table selected. The table has columns: product_name, price, and increment_price. The data consists of 10 rows, each representing a product with its name, original price, and the price increased by 10%. The table is displayed in a grid format with a light gray background and white text. The 'Result Grid' tab is active, and there are buttons for 'Filter Rows', 'Export', and 'Wrap Cell Content'.

	product_name	price	increment_price
▶	iPhone 15	79999.00	87998.9000
	Samsung Galaxy S24	74999.00	82498.9000
	Noise Smartwatch	4999.00	5498.9000
	Boat Earbuds	2499.00	2748.9000
	Kurta Set	1999.00	2198.9000
	Running Shoes	2999.00	3298.9000
	Prestige Mixer Grinder	6499.00	7148.9000
	Sony Bravia 55" TV	59999.00	65998.9000
	Lenovo Laptop	55999.00	61598.9000
	Philips Trimmer	1499.00	1648.9000

10. Show only the unique cities where customers live.

```
63      -- 10. Show only the unique cities where customers live
64 •  Select Distinct city from Customers;
65
```



city
Hyderabad
Mumbai
Ahmedabad
Chennai
Pune
Bengaluru
Kolkata
Lucknow
Bhubaneswar
Jaipur
Delhi
Noida
Indore

11. Get the first 3 customers who signed up.

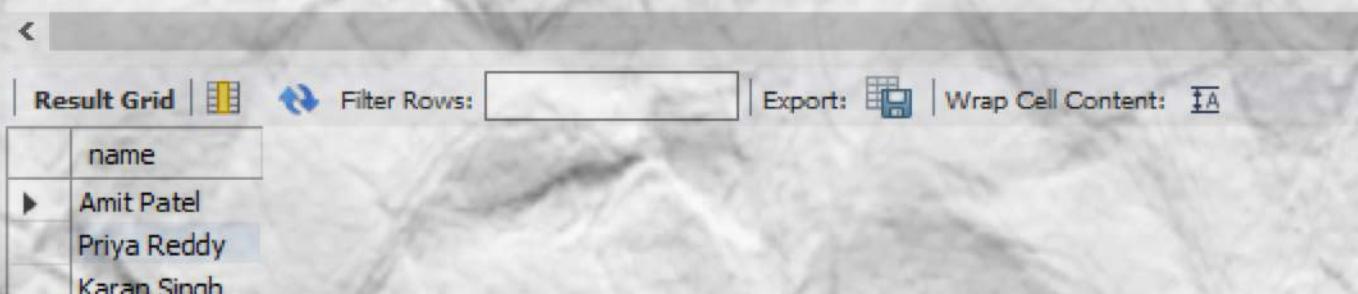
```
66      -- 11. Get the first 3 customers who signed up.
67 •  Select name from Customers where customer_id<4;
68
```



name
Rohit Kumar
Sneha Sharma
Amit Patel

12. Skip the first 2 customers and fetch the next 3 customers.

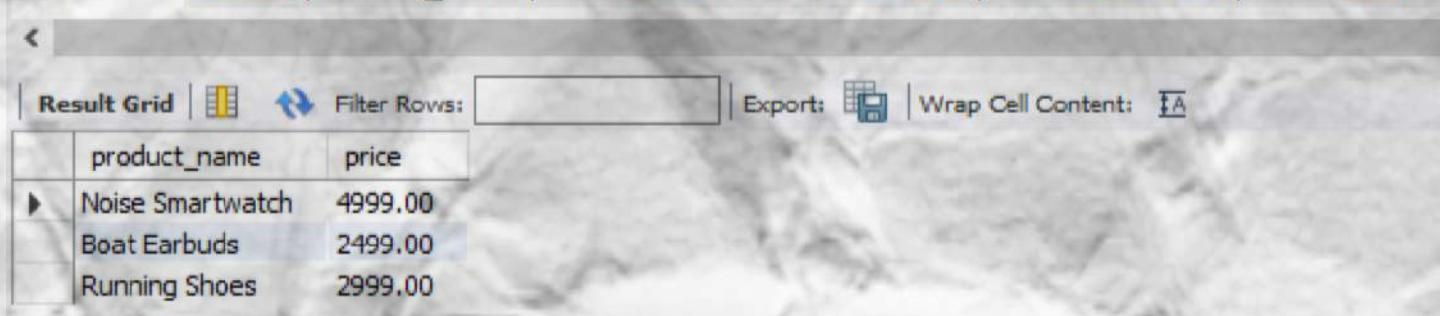
```
68
69      -- 12. Skip the first 2 customers and fetch the next 3 customers.
70 •  Select name from Customers where customer_id>=3 and customer_id<=5;
71
```



name
Amit Patel
Priya Reddy
Karan Singh

13. Find products with prices between ₹2000 and ₹6000.

```
72      -- 13. Find products with prices between ₹2000 and ₹6000.  
73 •  Select product_name, price from Products where price > 2000 and price < 6000;
```

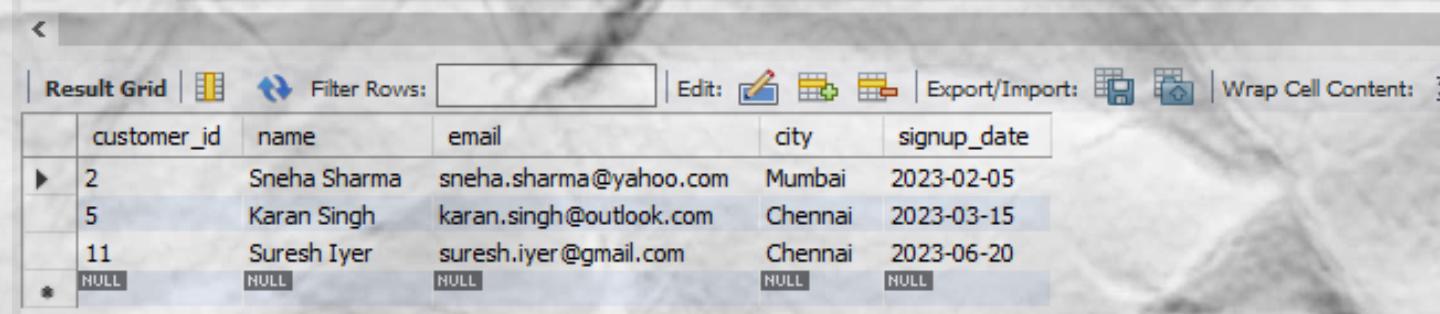


The screenshot shows a MySQL Workbench result grid with the following data:

	product_name	price
▶	Noise Smartwatch	4999.00
	Boat Earbuds	2499.00
	Running Shoes	2999.00

14. Find customers who are from Mumbai OR Chennai.

```
75      -- 14. Find customers who are from Mumbai OR Chennai.  
76 •  Select * from Customers where city = "Mumbai" or city = "Chennai";
```

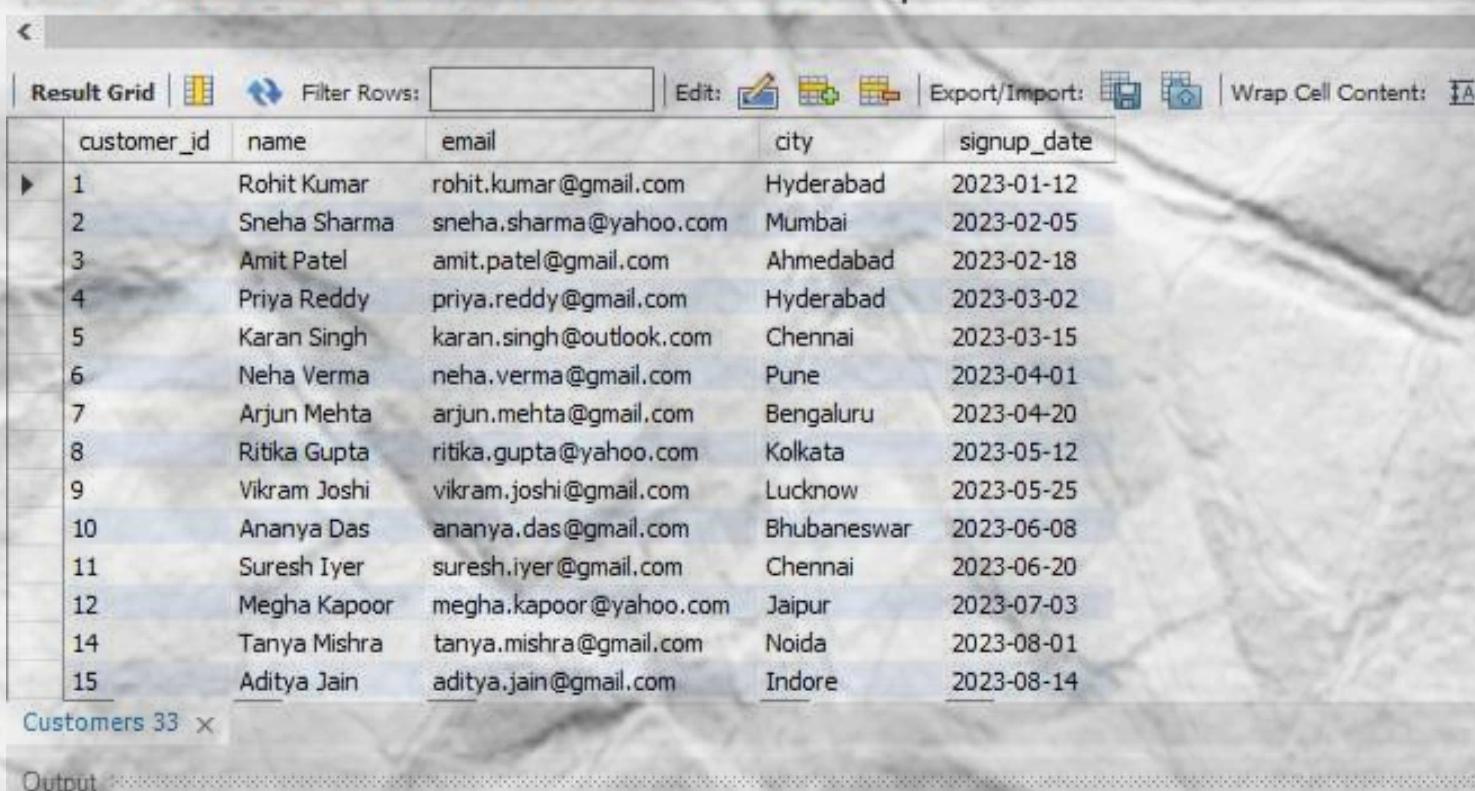


The screenshot shows a MySQL Workbench result grid with the following data:

	customer_id	name	email	city	signup_date
▶	2	Sneha Sharma	sneha.sharma@yahoo.com	Mumbai	2023-02-05
	5	Karan Singh	karan.singh@outlook.com	Chennai	2023-03-15
	11	Suresh Iyer	suresh.iyer@gmail.com	Chennai	2023-06-20
*	NULL	NULL	NULL	NULL	NULL

15. Find customers who are NOT from Delhi.

```
78      -- 15. Find customers who are NOT from Delhi.  
79 •  Select * from Customers where not city = "Delhi";
```

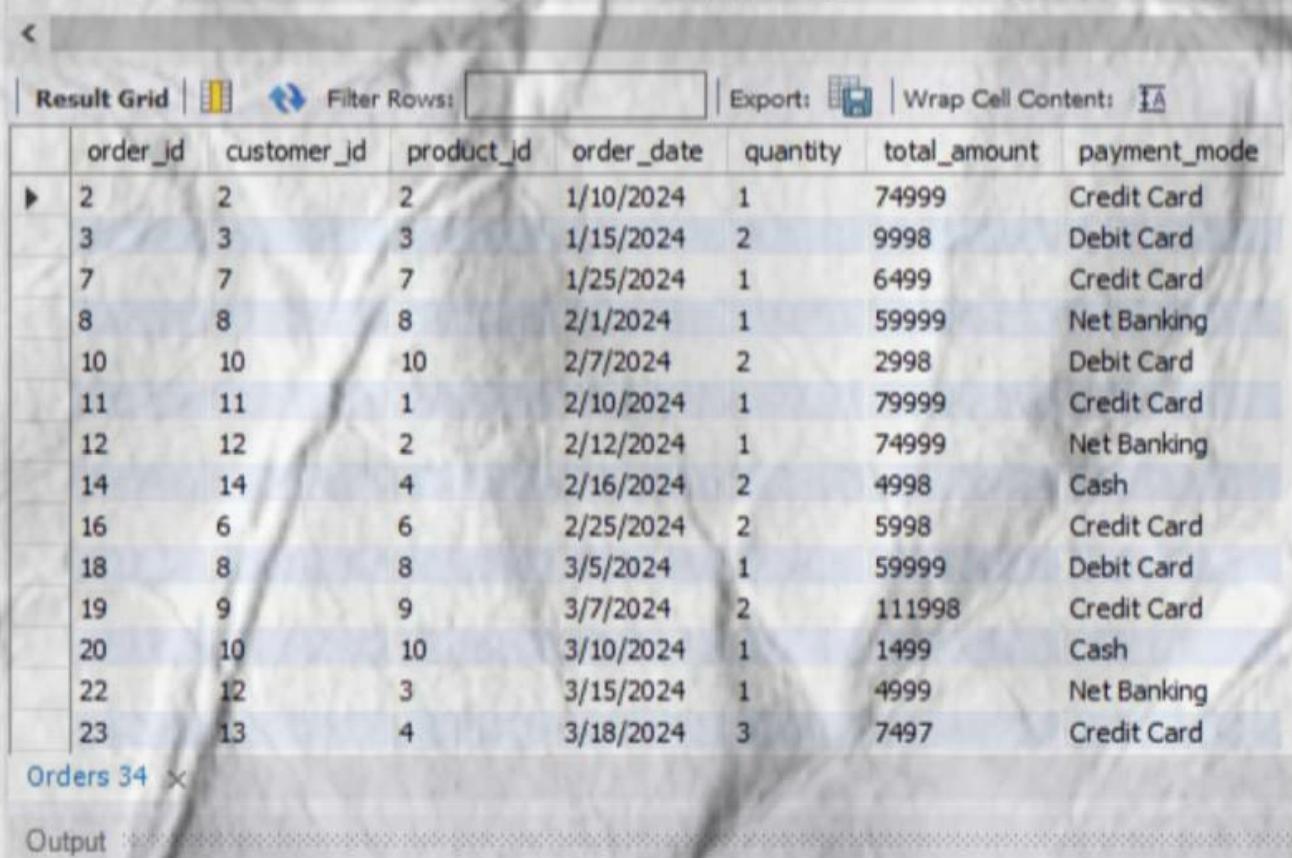


The screenshot shows a MySQL Workbench result grid with the following data:

	customer_id	name	email	city	signup_date
▶	1	Rohit Kumar	rohit.kumar@gmail.com	Hyderabad	2023-01-12
	2	Sneha Sharma	sneha.sharma@yahoo.com	Mumbai	2023-02-05
	3	Amit Patel	amit.patel@gmail.com	Ahmedabad	2023-02-18
	4	Priya Reddy	priya.reddy@gmail.com	Hyderabad	2023-03-02
	5	Karan Singh	karan.singh@outlook.com	Chennai	2023-03-15
	6	Neha Verma	neha.verma@gmail.com	Pune	2023-04-01
	7	Arjun Mehta	arjun.mehta@gmail.com	Bengaluru	2023-04-20
	8	Ritika Gupta	ritika.gupta@yahoo.com	Kolkata	2023-05-12
	9	Vikram Joshi	vikram.joshi@gmail.com	Lucknow	2023-05-25
	10	Ananya Das	ananya.das@gmail.com	Bhubaneswar	2023-06-08
	11	Suresh Iyer	suresh.iyer@gmail.com	Chennai	2023-06-20
	12	Megha Kapoor	megha.kapoor@yahoo.com	Jaipur	2023-07-03
	14	Tanya Mishra	tanya.mishra@gmail.com	Noida	2023-08-01
	15	Aditya Jain	aditya.jain@gmail.com	Indore	2023-08-14

16. Find orders that are NOT paid by UPI.

```
81      -- 16. Find orders that are NOT paid by UPI.  
82 •  Select * from Orders where not payment_mode="UPI";
```



	order_id	customer_id	product_id	order_date	quantity	total_amount	payment_mode
▶	2	2	2	1/10/2024	1	74999	Credit Card
	3	3	3	1/15/2024	2	9998	Debit Card
	7	7	7	1/25/2024	1	6499	Credit Card
	8	8	8	2/1/2024	1	59999	Net Banking
	10	10	10	2/7/2024	2	2998	Debit Card
	11	11	1	2/10/2024	1	79999	Credit Card
	12	12	2	2/12/2024	1	74999	Net Banking
	14	14	4	2/16/2024	2	4998	Cash
	16	6	6	2/25/2024	2	5998	Credit Card
	18	8	8	3/5/2024	1	59999	Debit Card
	19	9	9	3/7/2024	2	111998	Credit Card
	20	10	10	3/10/2024	1	1499	Cash
	22	12	3	3/15/2024	1	4999	Net Banking
	23	13	4	3/18/2024	3	7497	Credit Card

17. Get the average order amount across all orders.

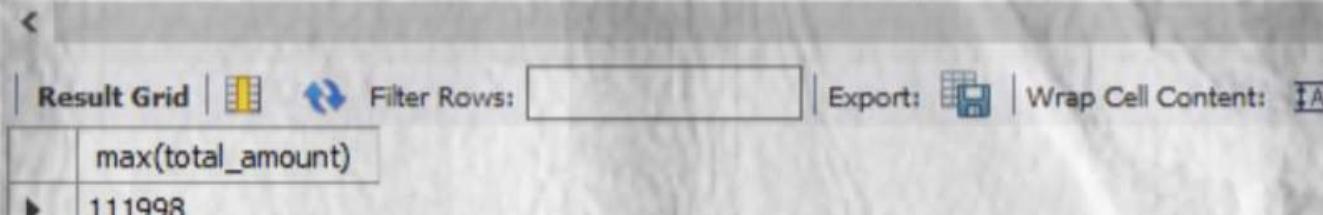
```
84      -- 17. Get the average order amount across all orders.  
85 •  Select sum(total_amount)/count(order_id) as average_amount from Orders;
```



	average_amount
▶	32722.8276

18. Show the highest order amount.

```
87      -- 18. Show the highest order amount.  
88 •  select max(total_amount) from Orders;
```



	max(total_amount)
▶	111998

19. Show the lowest product price.

```
90      -- 19. Show the lowest product price
91 •  select min(price) from Products;
```

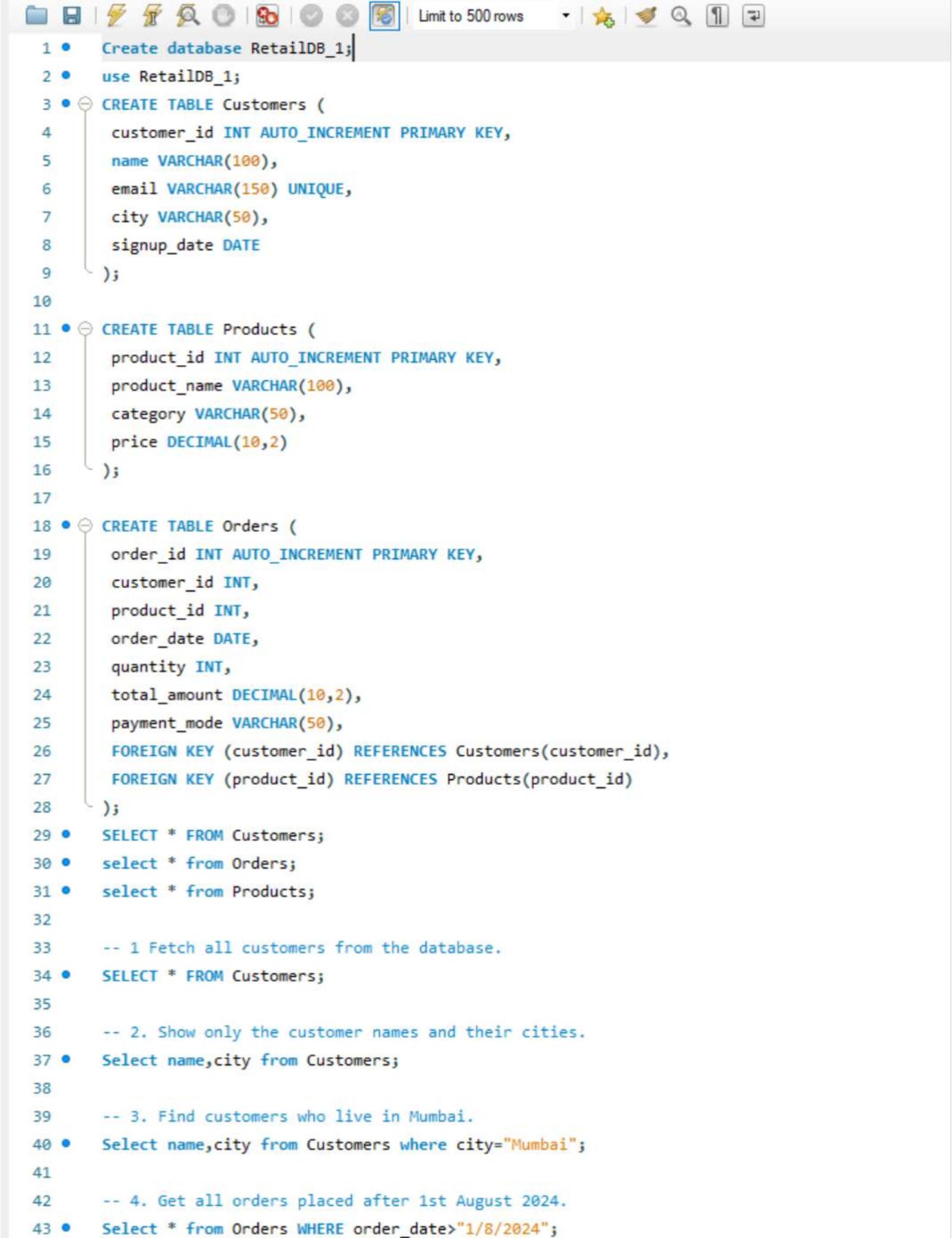
Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	min(price)			
▶	1499.00			

20. Find the total money spent across all orders

```
92
93      -- 20. Find the total money spent across all orders.
94 •  Select sum(total_amount) from orders;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	sum(total_amount)			
▶	948962			

Overview of MySQL workbench



The screenshot shows the MySQL Workbench interface with a script editor containing the following SQL code:

```
1 • Create database RetailDB_1;
2 • use RetailDB_1;
3 • CREATE TABLE Customers (
4     customer_id INT AUTO_INCREMENT PRIMARY KEY,
5     name VARCHAR(100),
6     email VARCHAR(150) UNIQUE,
7     city VARCHAR(50),
8     signup_date DATE
9 );
10
11 • CREATE TABLE Products (
12     product_id INT AUTO_INCREMENT PRIMARY KEY,
13     product_name VARCHAR(100),
14     category VARCHAR(50),
15     price DECIMAL(10,2)
16 );
17
18 • CREATE TABLE Orders (
19     order_id INT AUTO_INCREMENT PRIMARY KEY,
20     customer_id INT,
21     product_id INT,
22     order_date DATE,
23     quantity INT,
24     total_amount DECIMAL(10,2),
25     payment_mode VARCHAR(50),
26     FOREIGN KEY (customer_id) REFERENCES Customers(customer_id),
27     FOREIGN KEY (product_id) REFERENCES Products(product_id)
28 );
29 • SELECT * FROM Customers;
30 • select * from Orders;
31 • select * from Products;
32
33 -- 1 Fetch all customers from the database.
34 • SELECT * FROM Customers;
35
36 -- 2. Show only the customer names and their cities.
37 • Select name,city from Customers;
38
39 -- 3. Find customers who live in Mumbai.
40 • Select name,city from Customers where city="Mumbai";
41
42 -- 4. Get all orders placed after 1st August 2024.
43 • Select * from Orders WHERE order_date>"1/8/2024";
```

```
45      -- 5. List all products priced greater than ₹5000.
46 •  Select product_name, price from Products where price >5000;
47
48      -- 6. Count how many customers exist in the system.
49 •  Select count(customer_id) from Customers;
50
51      -- 7. Update a customer's city (e.g., change Rohit Kumar's city to Hyderabad).
52 •  set sql_safe_updates=0;
53 •  update Customers set city="Hyderabad" where name="Rohit Kumar";
54 •  select * from Customers;
55
56      -- 8. Delete an order (e.g., remove order with ID = 5).
57 •  delete from Orders where order_id=5;
58 •  select * from Orders;
59
60      -- 9. Display product names with their original price and price increased by 10%.
61 •  Select product_name,price,(price*1.10) as increment_price from Products;
62
63      -- 10. Show only the unique cities where customers live
64 •  Select Distinct city from Customers;
65
66      -- 11. Get the first 3 customers who signed up.
67 •  Select name from Customers where customer_id<4;
68
69      -- 12. Skip the first 2 customers and fetch the next 3 customers.
70 •  Select name from Customers where customer_id>=3 and customer_id<=5;
71
72      -- 13. Find products with prices between ₹2000 and ₹6000.
73 •  Select product_name,price from Products where price>2000 and price<6000;
74
75      -- 14. Find customers who are from Mumbai OR Chennai.
76 •  Select * from Customers where city="Mumbai" or city="Chennai";
77
78      -- 15. Find customers who are NOT from Delhi.
79 •  Select * from Customers where not city="Delhi";
80
81      -- 16. Find orders that are NOT paid by UPI.
82 •  Select * from Orders where not payment_mode="UPI";
83
84      -- 17. Get the average order amount across all orders.
85 •  Select sum(total_amount)/count(order_id) as average_amount from Orders;
86
87      -- 18. Show the highest order amount.
88 •  select max(total_amount) from Orders;
89
90      -- 19. Show the lowest product price
91 •  select min(price) from Products;
92
93      -- 20. Find the total money spent across all orders.
94 •  Select sum(total_amount) from orders;
```

Thank
you