

# Online Shopping Platform



-By Akshay Gujar  
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 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 3: Import data from CSV file

The screenshot shows the SQL Studio interface. On the left, the 'SCHEMAS' pane shows a database named 'retaildb\_1' with a table named 'customers'. A right-click context menu is open over the 'customers' table, and the 'Table Data Import Wizard' option is highlighted. The main SQL editor shows the following 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 );
```

Below the SQL editor, the 'Action Output' pane shows a successful execution of a query: 'select max(total\_amount) from Orders LIMIT 0, 500'.

## Step 4: Display Imported data

The screenshot shows the SQL Studio interface with the query 'SELECT \* FROM Customers;' executed. The results are displayed in a grid format. The grid has 5 columns: 'customer\_id', 'name', 'email', 'city', and 'signup\_date'. There are 15 rows of data, plus a final row with NULL values.

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

The screenshot shows a database management tool interface. On the left, a tree view shows the 'retaildb\_1' database with tables 'customers', 'orders', 'products', 'views', 'stored', and 'functions'. The 'orders' table is selected. A context menu is open over the 'orders' table, with 'Table Data Import Wizard' highlighted. The menu also includes options like 'Select Rows - Limit 500', 'Table Inspector', 'Copy to Clipboard', 'Table Data Export Wizard', 'Send to SQL Editor', 'Create Table...', 'Create Table Like...', 'Alter Table...', 'Table Maintenance...', 'Drop Table...', 'Truncate Table...', 'Search Table Data...', and 'Refresh All'. On the right, the SQL editor shows the following 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     phone VARCHAR(50),  
8     date DATE  
9 );  
10  
11 CREATE TABLE Products (  
12     product_id INT AUTO_INCREMENT PRIMARY KEY,  
13     name VARCHAR(100),  
14     description VARCHAR(50),  
15     price DECIMAL(10,2)
```

At the bottom, the 'Action Output' pane shows a successful query result:

#	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	1/5/2024	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
	5	5	5	1/20/2024	3	5997	Cash
	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
	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



## 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 SQL Studio interface. On the left, the 'SCHEMAS' pane shows a database named 'retaildb\_1' with tables 'customers', 'orders', and 'prod'. The 'prod' table is selected. A context menu is open over the 'prod' table, with 'Table Data Import Wizard' highlighted. The main SQL editor on the right shows 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     customer_name VARCHAR(150) UNIQUE,  
6     email VARCHAR(50),  
7     phone VARCHAR(20),  
8     address VARCHAR(255),  
9     created_at DATE  
10 );  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000
```

## Step 10: Display Imported data

```
31 • select * from 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	Sony Bravia 55" TV	Electronics	59999.00
	9	Lenovo Laptop	Electronics	55999.00
	10	Philips Trimmer	Personal Care	1499.00
*	NULL	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
```

<

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

Customers 12 ×

Output

## 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;
```

<

Result Grid | Filter Rows:  | Export: | Wrap Cell Content:

	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

Customers 13 ×

Output



3. Find customers who live in Mumbai.

39

-- 3. Find customers who live in Mumbai.

40

•

Select name,city from Customers where city="Mumbai";

<

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	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";

<

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	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;
```

Result Grid

Filter Rows:

Export

Wrap Cell Content:

	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;
```

Result Grid

Filter Rows:

Export

Wrap Cell Content:

	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;
```

Result Grid

Filter Rows:

Edit

Export/Import

Wrap Cell Content:

	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;

<

Result Grid

Filter Rows:

Export:

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

Orders 23 ×

Output

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

<

Result Grid

Filter Rows:

Export:

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

Result 24 ×




Output



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
```

<

Result Grid |  Filter Rows:  | Export:  | Wrap Cell Content: 

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




Customers 25 ×

Output

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
```

<




Result Grid |  Filter Rows:  | Export:  | Wrap Cell Content: 

	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
```

<

Result Grid |  Filter Rows:  | Export:  | Wrap Cell Content: 

	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;
```

Result Grid	Filter Rows:	Exports:	Wrap Cell Content:
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";
```

Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
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";
```

Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
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

Customers 33 x

Output



## 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";
```

<

Result Grid | Filter Rows:  | Export: | Wrap Cell Content:

	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

Orders 34 ×

Output

## 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;
```

<

Result Grid | Filter Rows:  | Export: | Wrap Cell Content:

	average_amount
▶	32722.8276

## 18. Show the highest order amount.

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

<

Result Grid | Filter Rows:  | Export: | Wrap Cell Content:

	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 displays the MySQL Workbench SQL editor interface. The top toolbar includes icons for file operations, execution, and navigation. A status bar at the top right indicates 'Limit to 500 rows'. The SQL editor contains the following 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