

```
108
109 •  SELECT e.*
110   FROM Events e
111   JOIN Venues v ON e.venue_id = v.venue_id
112   WHERE v.location = 'Ahmedabad'
113   AND e.event_date > CURDATE();
114
```

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

	event_id	event_name	event_date	venue_id	organizer_id	ticket_price	total_seats	available_seats
▶	1	Music Night	2026-03-10	1	1	600.00	500	450

```
114  
115 • SELECT e.event_id, e.event_name,  
116     SUM(p.amount_paid) AS total_revenue  
117 FROM Events e  
118 JOIN Tickets t ON e.event_id = t.event_id  
119 JOIN Payments p ON t.ticket_id = p.ticket_id  
120 WHERE p.payment_status = 'Success'  
...  
.....
```

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

	event_id	event_name	total_revenue
▶	5	Comedy Show	700.00
	1	Music Night	500.00
	3	Food Festival	300.00

```
5 • SELECT DISTINCT a.*  
6 FROM Attendees a  
7 JOIN Tickets t ON a.attendee_id = t.attendee_id  
8 WHERE t.booking_date >= CURDATE() - INTERVAL 7 DAY;
```

Attendee ID	Name	Email	Phone Number

- Apply SQL operators (AND, OR, Not) - Everything, if there is no selection

```
SELECT *
FROM Events
WHERE MONTH(event_date) = 12
AND available_seats > (total_seats * 0.5);
```

Event Grid				Filter Rows:				Export/Import:			Wrap Cell Content
event_id	event_name	event_date	venue_id	organizer_id	ticket_price	total_seats	available_seats				
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL				

```
• SELECT DISTINCT a.*  
FROM Attendees a  
LEFT JOIN Tickets t ON a.attendee_id = t.attendee_id  
LEFT JOIN Payments p ON t.ticket_id = p.ticket_id  
WHERE t.ticket_id IS NOT NULL
```

Full Grid			
attendee_id	name	email	phone_number
1	Jay	jay@gmail.com	9090909090
2	Rahul	rahul@gmail.com	8080808080
3	Priya	priya@gmail.com	7070707070
4	Neha	neha@gmail.com	6060606060
5	Amit	amit@gmail.com	5050505050

```
141     WHERE t.ticket_id IS NOT NULL  
142     OR p.payment_status = 'Pending';  
143  
144 • select * from Events where available_seats > 0;  
145  
146 -- Sorting and Grouping data (Order by , group by)  
147
```

```
145  
146      -- Sorting and Grouping data (Order by , group by)  
147  
148 •   SELECT *  
149     FROM Events  
150     ORDER BY event date ASC;
```

Result Grid				Filter Rows:		Edit:		Export/Import:		Wrap Cell Content	
	event_id	event_name	event_date	venue_id	organizer_id	ticket_price	total_seats	available_seats			
▶	1	Music Night	2026-03-10	1	1	600.00	500	450			
	2	Tech Conference	2026-04-15	5	2	1500.00	400	380			
	3	Food Festival	2026-05-20	2	3	300.00	300	250			
	4	Business Summit	2026-06-05	3	4	2000.00	800	750			
	5	Comedy Show	2026-07-18	4	5	700.00	1000	900			
	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL			

```
150     ORDER BY event_date ASC;  
151  
152 •   SELECT e.event_name,  
153         COUNT(t.attendee_id) AS total_attendees  
154     FROM Events e  
155     LEFT JOIN Tickets t ON e.event_id = t.event_id  
156     GROUP BY e.event_name;  
---
```

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

	event_name	total_attendees
▶	Music Night	1
	Tech Conference	1
	Food Festival	1
	Business Summit	1
	Comedy Show	1

```
156 GROUP BY e.event_name;  
157  
158 • SELECT e.event_name,  
159             SUM(p.amount_paid) AS total_revenue  
160     FROM Events e  
161     JOIN Tickets t ON e.event_id = t.event_id  
162     JOIN Payments p ON t.ticket_id = p.ticket_id  
---
```

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

	event_name	total_revenue
▶	Music Night	500.00
	Food Festival	300.00
	Comedy Show	700.00

- ```
SELECT SUM(amount_paid) AS total_revenue
FROM Payments
WHERE payment_status = 'Success';
```

| total_revenue |
|---------------|
| 1500.00       |

```
71
72 • SELECT e.event_id, e.event_name,
73 COUNT(t.ticket_id) AS total_attendees
74 FROM Events e
75 LEFT JOIN Tickets t ON e.event_id = t.event_id
76 GROUP BY e.event_id, e.event_name
77 ORDER BY total_attendees DESC
```

---

result Grid | Filter Rows: \_\_\_\_\_ | Export: Wrap Cell Content: Fetch rows

| event_id | event_name  | total_attendees |
|----------|-------------|-----------------|
| 1        | Music Night | 1               |

```
178 LIMIT 1;
179
180 • SELECT AVG(ticket_price) AS average_ticket_price
181 FROM Events;
182
183 -- Establish primary key and foreign key

```

---

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

| average_ticket_price |
|----------------------|
| 1020.000000          |

```
191 • SELECT e.event_id, e.event_name, e.event_date,
192 v.venue_name, v.location, v.capacity
193 FROM Events e
194 INNER JOIN Venues v
195 ON e.venue_id = v.venue_id;
196
197 • SELECT DISTINCT a.attendee_id, a.name, a.email

```

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

|   | event_id | event_name      | event_date | venue_name     | location  | capacity |
|---|----------|-----------------|------------|----------------|-----------|----------|
| ▶ | 1        | Music Night     | 2026-03-10 | Grand Hall     | Ahmedabad | 500      |
|   | 2        | Tech Conference | 2026-04-15 | Conference Hub | Pune      | 400      |
|   | 3        | Food Festival   | 2026-05-20 | City Center    | Surat     | 300      |
|   | 4        | Business Summit | 2026-06-05 | Royal Palace   | Mumbai    | 800      |
|   | 5        | Comedy Show     | 2026-07-18 | Open Arena     | Delhi     | 1000     |

- `SELECT DISTINCT a.attendee_id, a.name, a.email`  
Execute the selected portion of the script or everything, if there is no selection  
`FROM Attendees a`  
`LEFT JOIN Tickets t ON a.attendee_id = t.attendee_id`  
`LEFT JOIN Payments p ON t.ticket_id = p.ticket_id`  
`WHERE t.ticket_id IS NOT NULL`  
`AND (p.payment_status IS NULL OR p.payment_status <> 'Success');`

| Result Grid |       |                 | Filter Rows: | Export: | Wrap Cell Content: |
|-------------|-------|-----------------|--------------|---------|--------------------|
| attendee_id | name  | email           |              |         |                    |
| 2           | Rahul | rahul@gmail.com |              |         |                    |
| 4           | Neha  | neha@gmail.com  |              |         |                    |

203

204 • SELECT e.event\_id, e.event\_name  
205 FROM Tickets t  
206 RIGHT JOIN Events e  
207 ON t.event\_id = e.event\_id  
208 WHERE t.ticket\_id IS NULL;

209

Result Grid



Filter Rows:

Export:



Wrap Cell Content:

| event_id | event_name |
|----------|------------|
|----------|------------|

The screenshot shows a MySQL Workbench interface. The top bar includes standard file and database management icons. Below the toolbar, there's a dropdown menu labeled "Limit to 1000 rows". The main area contains a query editor with the following SQL code:

```
210 • SELECT a.attendee_id, a.name
211 FROM Attendees a
212 LEFT JOIN Tickets t
213 ON a.attendee_id = t.attendee_id
214 WHERE t.ticket_id IS NULL;
215
216 -- Use subqueries

```

Below the query editor is a horizontal toolbar with "Result Grid" selected, followed by "Filter Rows:" and "Export:" buttons. The "Result Grid" section displays the query results:

| attendee_id | name |
|-------------|------|
|-------------|------|



Limit to 1000 rows

```
218 • SELECT e.event_id, e.event_name,
219 SUM(p.amount_paid) AS total_revenue
220 FROM Events e
221 JOIN Tickets t ON e.event_id = t.event_id
222 JOIN Payments p ON t.ticket_id = p.ticket_id
223 WHERE p.payment_status = 'Success'
224 GROUP BY e.event_id, e.event_name
```

Result Grid



Filter Rows:

Export:



Wrap Cell Content

|   | event_id | event_name  | total_revenue |
|---|----------|-------------|---------------|
| ▶ | 5        | Comedy Show | 700.00        |

Limit to 1000 rows

```
• SELECT a.attendee_id, a.name,
 COUNT(DISTINCT t.event_id) AS total_events
 FROM Attendees a
 JOIN Tickets t ON a.attendee_id = t.attendee_id
 GROUP BY a.attendee_id, a.name
 HAVING COUNT(DISTINCT t.event_id) > 1;
```

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

| attendee_id | name | total_events |
|-------------|------|--------------|
|-------------|------|--------------|

```
236
237 • SELECT o.organizer_id, o.organizer_name,
238 COUNT(e.event_id) AS total_events_managed
239 FROM Organizers o
240 JOIN Events e ON o.organizer_id = e.organizer_id
```

---

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

| organizer_id | organizer_name | total_events_managed |
|--------------|----------------|----------------------|
|--------------|----------------|----------------------|

```
246 • SELECT event_id, event_name,
247 MONTH(event_date) AS event_month
248 FROM Events;
249
250 • SELECT event_id, event_name, event_date,
251 DATEDIFF(event_date, CURDATE()) AS days_remaining
252 FROM Events
```

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

|   | event_id        | event_name | event_month |
|---|-----------------|------------|-------------|
| 1 | Music Night     | 3          |             |
| 2 | Tech Conference | 4          |             |
| 3 | Food Festival   | 5          |             |
| 4 | Business Summit | 6          |             |
| 5 | Comedy Show     | 7          |             |

```
249
250 • SELECT event_id, event_name, event_date,
251 DATEDIFF(event_date, CURDATE()) AS days_remaining
252 FROM Events
253 WHERE event_date >= CURDATE();
254
255 • SELECT payment_id,
```

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

|   | event_id | event_name      | event_date | days_remaining |
|---|----------|-----------------|------------|----------------|
| • | 1        | Music Night     | 2026-03-10 | 13             |
|   | 2        | Tech Conference | 2026-04-15 | 49             |
|   | 3        | Food Festival   | 2026-05-20 | 84             |
|   | 4        | Business Summit | 2026-06-05 | 100            |
|   | 5        | Comedy Show     | 2026-07-18 | 143            |

```
255 • SELECT payment_id,
256 DATE_FORMAT(payment_date, '%Y-%m-%d %H:%i:%s') AS formatted_payment_date
257 FROM Payments;
258
259 -- String manipulation function
260
261 • SELECT organizer_id,

```

---

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

|   | payment_id | formatted_payment_date |
|---|------------|------------------------|
| ▶ | 1          | 2026-02-01 00:00:00    |
|   | 2          | 2026-02-02 00:00:00    |
|   | 3          | 2026-02-03 00:00:00    |
|   | 5          | 2026-02-05 00:00:00    |

```
259 -- String manipulation function
260
261 • SELECT organizer_id,
262 UPPER(organizer_name) AS organizer_name_uppercase
263 FROM Organizers;
264
```

---

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

| organizer_id | organizer_name_uppercase |
|--------------|--------------------------|
| 1            | ABC EVENTS               |
| 2            | STAR PLANNERS            |
| 3            | ELITE ORG                |
| 4            | PRIME EVENTS             |
| 5            | MEGA SHOWS               |

```
263 FROM Organizers;
264
265 • SELECT attendee_id,
266 TRIM(name) AS cleaned_name
267 FROM Attendees;
```

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

| attendee_id | cleaned_name |
|-------------|--------------|
| 1           | Jay          |
| 2           | Rahul        |
| 3           | Priya        |
| 4           | Neha         |
| 5           | Amit         |

```
268
269 • SELECT attendee_id,
270 name,
271 COALESCE(email, 'Not Provided') AS email_status
272 FROM Attendees;
273
```

---

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

|   | attendee_id | name  | email_status    |
|---|-------------|-------|-----------------|
| ▶ | 1           | Jay   | jay@gmail.com   |
|   | 2           | Rahul | rahul@gmail.com |
|   | 3           | Priya | priya@gmail.com |
|   | 4           | Neha  | neha@gmail.com  |
|   | 5           | Amit  | amit@gmail.com  |

```
276 • SELECT event_name,
277 SUM(p.amount_paid) AS total_revenue,
278 RANK() OVER (ORDER BY SUM(p.amount_paid) DESC) AS revenue_rank
279 FROM Events e
280 JOIN Tickets t ON e.event_id = t.event_id
281 JOIN Payments p ON t.ticket_id = p.ticket_id
282 WHERE p.payment_status = 'Success'
```

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

| event_name    | total_revenue | revenue_rank |
|---------------|---------------|--------------|
| Comedy Show   | 700.00        | 1            |
| Music Night   | 500.00        | 2            |
| Food Festival | 300.00        | 3            |

```
85 • SELECT e.event_name,
86 SUM(p.amount_paid) AS total_sales,
87 SUM(SUM(p.amount_paid)) OVER (ORDER BY e.event_date) AS cumulative_sales
88 FROM Events e
89 JOIN Tickets t ON e.event_id = t.event_id
90 JOIN Payments p ON t.ticket_id = p.ticket_id
91 WHERE p.payment_status = 'Success'
```

| Result Grid   |             |                  |
|---------------|-------------|------------------|
| event_name    | total_sales | cumulative_sales |
| Music Night   | 500.00      | 500.00           |
| Food Festival | 300.00      | 800.00           |
| Comedy Show   | 700.00      | 1500.00          |

```
294 • SELECT e.event_name,
295 t.booking_date,
296 COUNT(t.ticket_id) OVER (
297 PARTITION BY e.event_id
298 ORDER BY t.booking_date
299) AS running_total_attendees
300 FROM Events e
```

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

|   | event_name      | booking_date | running_total_attendees |
|---|-----------------|--------------|-------------------------|
| ▶ | Music Night     | 2026-02-01   | 1                       |
|   | Tech Conference | 2026-02-02   | 1                       |
|   | Food Festival   | 2026-02-03   | 1                       |
|   | Business Summit | 2026-02-04   | 1                       |
|   | Comedy Show     | 2026-02-05   | 1                       |

```
total_seats,
available_seats,
CASE
 WHEN available_seats < (0.2 * total_seats)
 THEN 'High Demand'
 WHEN available_seats BETWEEN (0.2 * total_seats)
 AND (0.5 * total_seats)
```

| Result Grid     |             |                 |                 |
|-----------------|-------------|-----------------|-----------------|
| event_name      | total_seats | available_seats | demand_category |
| Music Night     | 500         | 450             | Low Demand      |
| Tech Conference | 400         | 380             | Low Demand      |
| Food Festival   | 300         | 250             | Low Demand      |
| Business Summit | 800         | 750             | Low Demand      |
| Comedy Show     | 1000        | 900             | Low Demand      |

```
316 FROM Events;
317
318 • SELECT payment_id,
319 payment_status,
320 CASE
321 WHEN payment_status = 'Success' THEN 'Successful'
322 WHEN payment_status = 'Failed' THEN 'Failed'
323 END AS payment_result
324
```

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

|   | payment_id | payment_status | payment_result |
|---|------------|----------------|----------------|
| ▶ | 1          | Success        | Successful     |
|   | 2          | Pending        | Pending        |
|   | 3          | Success        | Successful     |
|   | 5          | Success        | Successful     |