

INPUT

168







169 -- 1. Basic SELECT Queries

170

171 -- All customers

172 ● SELECT * FROM customers;

OUTPUT

Result Grid					
Filter Rows: <input type="text"/>					
Edit:   					
Export/Import:  					
Wrap Cell Content: 					
	customer_id	name	email	phone	address
1		Alice Johnson	alice@gmail.com	9876543210	New York
2		Bob Smith	bob@gmail.com	8765432109	California
3		Charlie Lee	charlie@gmail.com	7654321098	Texas
4		David Kim	david.kim@gmail.com	7543210987	Washington
5		Evelyn Clark	evelyn.clark@gmail.com	7432109876	Florida
6		Frank Wright	frank.wright@gmail.com	7321098765	Illinois
7		Grace Hall	grace.hall@gmail.com	7210987654	Ohio
8		Hannah Adams	hannah.adams@gmail.com	7109876543	Georgia
9		Ian Baker	ian.baker@gmail.com	7098765432	Nevada
10		Jackie Chen	jackie.chen@gmail.com	6987654321	Oregon
11		Karen Davis	karen.davis@gmail.com	6876543210	Colorado
12		Liam Evans	liam.evans@gmail.com	6765432109	Michigan
13		Mia Foster	mia.foster@gmail.com	6654321098	Arizona
14		Nathan Gray	nathan.gray@gmail.com	6543210987	Indiana
15		Olivia Harris	olivia.harris@gmail.com	6432109876	Tennessee
16		Paul Irving	paul.irving@gmail.com	6321098765	Missouri
17		Queenie Jones	queenie.jones@gmail.com	6210987654	Louisiana
18		Ryan King	ryan.king@gmail.com	6109876543	Kentucky
19		Sophia Lee	sophia.lee@gmail.com	6098765432	Alabama
20		Tommy Nelson	tommy.nelson@gmail.com	5987654321	Massachus...
	NULL	NULL	NULL	NULL	NULL

INPUT









173

174 -- All orders placed in May 2025

175 ● SELECT * FROM orders WHERE order_date BETWEEN '2025-05-01' AND '2025-05-31';

176

OUTPUT

<				
Result Grid		 Filter Rows: <input type="text"/>	Edit:   	Export/Import:   Wrap Cell Content: 
	order_id	customer_id	order_date	total_amount
▶	201	1	2025-05-01	799.99
	202	2	2025-05-02	149.99
	203	3	2025-05-03	999.99
	204	4	2025-05-03	349.99
	205	5	2025-05-04	180.00
	206	6	2025-05-04	59.99
	207	7	2025-05-05	129.95
	208	8	2025-05-05	549.99
	209	9	2025-05-06	49.99
	210	10	2025-05-06	139.99
	211	11	2025-05-07	119.95
	212	12	2025-05-08	649.99
	213	13	2025-05-08	150.00
	214	14	2025-05-09	59.99
	215	15	2025-05-09	999.99
	216	16	2025-05-10	399.99
	217	17	2025-05-10	849.50
	218	18	2025-05-11	39.99
	219	19	2025-05-12	150.00
	220	20	2025-05-12	799.99
•	NULL	NULL	NULL	NULL

INPUT

```
176
177      -- Products under ₹500
178 ●    SELECT * FROM products WHERE price < 500;
179
```

OUTPUT

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

	product_id	name	category	price	stock_quantity
▶	103	Sony WH-1000XM4 Headphones	Audio	349.99	30
	107	Apple Watch Series 8	Wearables	399.99	40
	108	Fitbit Charge 5	Wearables	129.95	25
	109	Amazon Echo Dot (5th Gen)	Smart Home	49.99	60
	110	Google Nest Mini	Smart Home	39.99	55
	111	Nike Air Max 270	Footwear	150.00	35
	112	Adidas Ultraboost 22	Footwear	180.00	28
	113	Logitech MX Master 3 Mouse	Accessories	99.99	75
	114	Razer BlackWidow Keyboard	Accessories	129.99	20
	117	Kindle Paperwhite	Books & Media	139.99	22
	118	JBL Flip 6 Bluetooth Speaker	Audio	119.95	33
	120	WD 1TB External HDD	Storage	59.99	65
✖	NULL	NULL	NULL	NULL	NULL

INPUT

179

180 -- 2. WHERE, ORDER BY, GROUP BY

181

182 -- Customers in California

183 ● SELECT * FROM customers WHERE address = 'California';

184

OUTPUT

<

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

	customer_id	name	email	phone	address
	2	Bob Smith	bob@gmail.com	8765432109	California
	NULL	NULL	NULL	NULL	NULL

INPUT

```
184
185      -- Orders sorted by amount
186 ●    SELECT * FROM orders ORDER BY total_amount DESC;
187
```

OUTPUT

<

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

	order_id	customer_id	order_date	total_amount
▶	203	3	2025-05-03	999.99
	215	15	2025-05-09	999.99
	217	17	2025-05-10	849.50
	201	1	2025-05-01	799.99
	220	20	2025-05-12	799.99
	212	12	2025-05-08	649.99
	208	8	2025-05-05	549.99
	216	16	2025-05-10	399.99
	204	4	2025-05-03	349.99
	205	5	2025-05-04	180.00
	213	13	2025-05-08	150.00
	219	19	2025-05-12	150.00
	202	2	2025-05-02	149.99
	210	10	2025-05-06	139.99
	207	7	2025-05-05	129.95
	211	11	2025-05-07	119.95
	206	6	2025-05-04	59.99
	214	14	2025-05-09	59.99
	209	9	2025-05-06	49.99
	218	18	2025-05-11	39.99
*	NULL	NULL	NULL	NULL

orders 40 ×

INPUT

```
188      -- Total orders per customer
189 •    SELECT customer_id, COUNT(*) AS total_orders
190      FROM orders
191      GROUP BY customer_id;
```

OUTPUT

<

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	customer_id	total_orders
▶	1	1
	2	1
	3	1
	4	1
	5	1
	6	1
	7	1
	8	1
	9	1
	10	1
	11	1
	12	1
	13	1
	14	1
	15	1
	16	1
	17	1
	18	1
	19	1
	20	1

INPUT

```
192
193      -- 3. JOINS: INNER, LEFT, RIGHT
194
195      -- INNER JOIN: Orders with customer names
196 •  SELECT o.order_id, c.name, o.total_amount
197      FROM orders o
198      INNER JOIN customers c ON o.customer_id = c.customer_id;
```

OUTPUT

<			
Result Grid			
Filter Rows:			
Export:			
Wrap Cell Content:			
	order_id	name	total_amount
▶	201	Alice Johnson	799.99
	202	Bob Smith	149.99
	203	Charlie Lee	999.99
	204	David Kim	349.99
	205	Evelyn Clark	180.00
	206	Frank Wright	59.99
	207	Grace Hall	129.95
	208	Hannah Adams	549.99
	209	Ian Baker	49.99
	210	Jackie Chen	139.99
	211	Karen Davis	119.95
	212	Liam Evans	649.99
	213	Mia Foster	150.00
	214	Nathan Gray	59.99
	215	Olivia Harris	999.99
	216	Paul Irving	399.99
	217	Queenie Jones	849.50
	218	Ryan King	39.99
	219	Sophia Lee	150.00
	220	Tommy Nelson	799.99


INPUT


```
199
200      -- LEFT JOIN: All customers even if they haven't ordered
201 ●    SELECT c.customer_id, c.name, o.order_id
202      FROM customers c
203      LEFT JOIN orders o ON c.customer_id = o.customer_id;
204
```

OUTPUT


<

Result Grid




 Filter Rows:

Export:



Wrap Cell Content:



	customer_id	name	order_id
▶	1	Alice Johnson	201
	2	Bob Smith	202
	3	Charlie Lee	203
	4	David Kim	204
	5	Evelyn Clark	205
	6	Frank Wright	206
	7	Grace Hall	207
	8	Hannah Adams	208
	9	Ian Baker	209
	10	Jackie Chen	210
	11	Karen Davis	211
	12	Liam Evans	212
	13	Mia Foster	213
	14	Nathan Gray	214
	15	Olivia Harris	215
	16	Paul Irving	216
	17	Queenie Jones	217
	18	Ryan King	218
	19	Sophia Lee	219
	20	Tommy Nelson	220


INPUT


```
208      -- 4. Aggregate Functions: SUM, AVG
209
210      -- Total sales amount
211 •    SELECT SUM(total_amount) AS total_sales FROM orders;
212
```

OUTPUT

<


Result Grid






Filter Rows:

Export:



Wrap Cell Content:



	total_sales
▶	7629.26

INPUT

```
213      -- Average order value
214 •    SELECT AVG(total_amount) AS avg_order_value FROM orders;
215
```

OUTPUT

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	avg_order_value
	381.463000

INPUT

```
216      -- Sales by payment method
217 •    SELECT payment_method, SUM(amount) AS total_by_method
218      FROM payments
219      GROUP BY payment_method;
220
```

OUTPUT

<

Result Grid

Filter Rows:

Export:







Wrap Cell Content:

	payment_method	total_by_method
▶	Credit Card	3989.85
	UPI	1849.48
	Net Banking	799.97
	Debit Card	869.98
	Cash on Delivery	119.98

INPUT

```
221
222     -- 5. Subqueries
223
224     -- Customers who made orders over ₹500
225 ●   SELECT * FROM customers
226  ⊖  WHERE customer_id IN (
227      SELECT customer_id FROM orders WHERE total_amount > 500
228  );
229
```

OUTPUT

Result Grid					
Filter Rows: <input type="text"/>					
Edit:   					
Export/Import:  					
Wrap Cell Content: 					
	customer_id	name	email	phone	address
1		Alice Johnson	alice@gmail.com	9876543210	New York
3		Charlie Lee	charlie@gmail.com	7654321098	Texas
8		Hannah Adams	hannah.adams@gmail.com	7109876543	Georgia
12		Liam Evans	liam.evans@gmail.com	6765432109	Michigan
15		Olivia Harris	olivia.harris@gmail.com	6432109876	Tennessee
17		Queenie Jones	queenie.jones@gmail.com	6210987654	Louisiana
20		Tommy Nelson	tommy.nelson@gmail.com	5987654321	Massachusetts
	NULL	NULL	NULL	NULL	NULL

INPUT

229

230 -- 6. Create Views for Reuse

231

232 -- View: Sales summary by customer

233 ● CREATE VIEW customer_sales_summary AS

234 SELECT c.customer_id, c.name, SUM(o.total_amount) AS total_spent

235 FROM customers c

236 JOIN orders o ON c.customer_id = o.customer_id

237 GROUP BY c.customer_id, c.name;

238

INPUT

```
239
240      -- 7. Optimize with Indexes
241
242      -- Improve search performance on frequently filtered columns
243 • CREATE INDEX idx_customer_id ON orders(customer_id);
244 • CREATE INDEX idx_order_date ON orders(order_date);
245
```

INPUT

```
246
247      -- 8. Real Sales Reports Examples
248
249      -- Sales by Product Category
250  ●  SELECT p.category, SUM(oi.price * oi.quantity) AS category_sales
251     FROM order_items oi
252     JOIN products p ON oi.product_id = p.product_id
253     GROUP BY p.category
254     ORDER BY category_sales DESC;
255
```

OUTPUT

251 FROM order_items oi

Result Grid



Filter Rows:

Export:



Wrap Cell Content:



	category	category_sales
▶	Computers	3499.47
	Electronics	2149.97
	Footwear	629.99
	Wearables	529.94
	Audio	469.94
	Books & Media	139.99
	Storage	119.98
	Smart Home	89.98

INPUT

```
255
256      -- Sales by Date
257 •    SELECT order_date, SUM(total_amount) AS daily_sales
258      FROM orders
259      GROUP BY order_date
260      ORDER BY order_date;
261
```

OUTPUT

Result Grid

Filter Rows:

Export:





Wrap Cell Content:

	order_date	daily_sales
▶	2025-05-01	799.99
	2025-05-02	149.99
	2025-05-03	1349.98
	2025-05-04	239.99
	2025-05-05	679.94
	2025-05-06	189.98
	2025-05-07	119.95
	2025-05-08	799.99
	2025-05-09	1059.98
	2025-05-10	1249.49
	2025-05-11	39.99
	2025-05-12	949.99

INPUT

```
261
262     -- Sales by Customer
263 •  SELECT c.name, SUM(o.total_amount) AS total_spent
264     FROM customers c
265     JOIN orders o ON c.customer_id = o.customer_id
266     GROUP BY c.name
267     ORDER BY total_spent DESC;
268
```

OUTPUT

Result Grid				 Filter Rows: <input type="text"/>	Export:  Wrap Cell Content: 
	name	total_spent			
▶	Charlie Lee	999.99			
	Olivia Harris	999.99			
	Queenie Jones	849.50			
	Alice Johnson	799.99			
	Tommy Nelson	799.99			
	Liam Evans	649.99			
	Hannah Adams	549.99			
	Paul Irving	399.99			
	David Kim	349.99			
	Evelyn Clark	180.00			
	Mia Foster	150.00			
	Sophia Lee	150.00			
	Bob Smith	149.99			
	Jackie Chen	139.99			
	Grace Hall	129.95			
	Karen Davis	119.95			
	Frank Wright	59.99			
	Nathan Gray	59.99			
	Ian Baker	49.99			
	Ryan King	39.99			