```sql
-- A. WHERE vs HAVING
-- WHERE filters rows before aggregation (orders in 2025)
SELECT * FROM orders
WHERE order_date BETWEEN '2025-01-01' AND '2025-12-31'
ORDER BY order_date DESC LIMIT 20;
```

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

order_id	user_id	order_date	status	payment_method
107	113	2025-11-16	delivered	debit_card
392	63	2025-11-04	delivered	net_banking
487	58	2025-10-28	shipped	credit_card
236	122	2025-10-28	delivered	upi
166	95	2025-10-27	delivered	credit_card
201	89	2025-10-26	processing	credit_card
199	32	2025-10-23	delivered	debit_card
181	128	2025-10-21	delivered	upi
521	134	2025-10-19	shipped	paypal
131	27	2025-10-14	processing	paypal

orders 12 ×

```sql
-- HAVING filters groups after aggregation (products with revenue > 2000)
SELECT p.product_id, p.product_name, p.category, SUM(oi.total_price) AS revenue
FROM products p
JOIN order_items oi ON p.product_id = oi.product_id
GROUP BY p.product_id, p.product_name, p.category
HAVING SUM(oi.total_price) > 2000
ORDER BY revenue DESC;
```

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

product_id	product_name	category	revenue
74	Clothing Item 74	Clothing	45971.48
20	Computers Item 20	Computers	34805.62
48	Toys Item 48	Toys	32933.70
10	Computers Item 10	Computers	31312.42
67	Electronics Item 67	Electronics	30147.32
72	Toys Item 72	Toys	29470.35
24	Beauty Item 24	Beauty	29060.25
73	Computers Item 73	Computers	28369.81
77	Beauty Item 77	Beauty	28293.66
70	Sports Item 70	Sports	28151.73

```sql
-- B. JOINS examples
-- INNER JOIN: users who placed orders and their order totals
SELECT u.user_id, u.name, o.order_id, ot.order_total
FROM users u
JOIN orders o ON u.user_id = o.user_id
JOIN order_totals_view ot ON o.order_id = ot.order_id
ORDER BY ot.order_total DESC LIMIT 20;
```

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

user_id	name	order_id	order_total
9	Arjun Brown	306	6472.95
84	Maya Mehra	220	6408.26
85	Maya Jones	33	6347.94
143	Rohit Kumar	324	5900.00
6	Priya Davis	30	5735.77
143	Rohit Kumar	330	5701.79
72	Rohit Gupta	231	5582.70
85	Maya Jones	346	5559.63
66	Sophia Smith	141	5555.12
82	Maya Singh	238	5364.76

```sql
SELECT u.user_id, u.name, ot.order_total, ot.order_date
FROM users u
LEFT JOIN (
    SELECT o.user_id, o.order_id, o.order_date, SUM(oi.total_price) AS order_total
    FROM orders o JOIN order_items oi ON o.order_id = oi.order_id
    GROUP BY o.order_id, o.user_id, o.order_date
) ot ON u.user_id = ot.user_id
GROUP BY u.user_id, ot.order_date, ot.order_total
ORDER BY u.user_id LIMIT 50;
```

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

user_id	name	order_total	order_date
1	Aditya Sharma	2822.42	2020-04-03
1	Aditya Sharma	882.04	2020-06-29
1	Aditya Sharma	952.26	2023-02-09
2	Priya Singh	1032.36	2021-10-08
2	Priya Singh	3782.87	2022-05-29
2	Priya Singh	686.37	2024-08-11
3	Sophia Jones	4105.62	2021-11-03
3	Sophia Jones	2155.45	2022-06-14
3	Sophia Jones	1667.28	2022-12-07
3	Sophia Jones	1258.64	2023-11-08

```sql
-- C. Average revenue per user (ARPU)
-- total revenue / count(distinct active users)
SELECT ROUND(SUM(oi.total_price) / NULLIF(COUNT(DISTINCT o.user_id),0),2) AS arpu_estimate
FROM orders o JOIN order_items oi ON o.order_id = oi.order_id;
```

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

arpu_estimate
6858.92

```sql
-- D. Subqueries examples
-- Users who spent more than average user
SELECT user_id, name, user_total FROM (
  SELECT u.user_id, u.name, COALESCE(SUM(oi.total_price),0) AS user_total
  FROM users u
  LEFT JOIN orders o ON u.user_id = o.user_id
  LEFT JOIN order_items oi ON o.order_id = oi.order_id
  GROUP BY u.user_id, u.name
) t
WHERE user_total > (
  SELECT AVG(user_total) FROM (
    SELECT COALESCE(SUM(oi.total_price),0) AS user_total
    FROM users u
```

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

user_id	name	user_total
3	Sophia Jones	9186.99
5	Isabella Sharma	7381.62
6	Priya Davis	6460.09
9	Arjun Brown	13434.46
12	Ava Nair	11115.73

```sql
-- E. Aggregations & business queries
-- Top 10 products by revenue
SELECT p.product_id, p.product_name, p.category, SUM(oi.total_price) AS revenue, SUM(oi.quantity) AS units_sold
FROM products p
JOIN order_items oi ON p.product_id = oi.product_id
GROUP BY p.product_id, p.product_name, p.category
ORDER BY revenue DESC LIMIT 10;
```

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

product_id	product_name	category	revenue	units_sold
74	Clothing Item 74	Clothing	45971.48	92
20	Computers Item 20	Computers	34805.62	71
48	Toys Item 48	Toys	32933.70	69
10	Computers Item 10	Computers	31312.42	71
67	Electronics Item 67	Electronics	30147.32	67
72	Toys Item 72	Toys	29470.35	85
24	Beauty Item 24	Beauty	29060.25	75
73	Computers Item 73	Computers	28369.81	67
77	Beauty Item 77	Beauty	28293.66	57
70	Sports Item 70	Sports	28151.73	57

```sql
-- Top 10 users by lifetime spend (from view)
SELECT user_id, name, lifetime_spend, orders_count
FROM customer_ltv_view
ORDER BY lifetime_spend DESC LIMIT 10;

```

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

user_id	name	lifetime_spend	orders_count
24	Sophia Brown	29805.87	9
85	Maya Jones	22390.09	7
130	Liam Smith	21992.61	8
143	Rohit Kumar	18899.56	6
93	Maya Brown	17955.13	6
158	Ananya Reddy	16512.47	8
35	Rohit Iyer	16460.54	6
184	Vihaan Iyer	15632.74	5
20	Sara Brown	15524.45	6
48	Aditya Jones	15371.82	7

```sql
-- G. Segmentation using CASE
SELECT user_id, name, lifetime_spend,
  CASE
    WHEN lifetime_spend >= 2000 THEN 'Platinum'
    WHEN lifetime_spend >= 1000 THEN 'Gold'
    WHEN lifetime_spend >= 500 THEN 'Silver'
    ELSE 'Bronze'
  END AS tier
```

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

user_id	name	lifetime_spend	tier
24	Sophia Brown	29805.87	Platinum
85	Maya Jones	22390.09	Platinum
130	Liam Smith	21992.61	Platinum
143	Rohit Kumar	18899.56	Platinum
93	Maya Brown	17955.13	Platinum
158	Ananya Reddy	16512.47	Platinum
35	Rohit Iyer	16460.54	Platinum
184	Vihaan Iyer	15632.74	Platinum
20	Sara Brown	15524.45	Platinum
48	Aditya Jones	15371.82	Platinum
82	Maya Singh	14889.08	Platinum