

Assignment Queries

```
4      -- 1) simple queries
5
6      -- 1. Retrieve all customers who have placed at least one order.
7 •    SELECT c.first_name, c.last_name FROM online_shopping_customers osc
8      JOIN citizens c ON c.id = osc.citizen_id
9      JOIN orders o ON o.customer_id = osc.id;
10
```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	first_name	last_name			
▶	Robert	Johnson			
	Jennifer	Johnson			
	David	Thompson			
	Emily	Thompson			
	Jessica	Martinez			
	Sophia	Lee			
	Matthew	Brown			
	William	Miller			
	Emma	Rodriguez			
	Andrew	Kim			

```
11     -- 2. Find students enrolled in a specific course.
12 •    SELECT c.first_name, c.last_name, cs.course_name FROM students s
13     JOIN citizens c ON c.id = s.citizen_id
14     JOIN enrollments e ON e.student_id = s.id
15     JOIN courses cs ON cs.id = e.course_id
16     WHERE cs.course_name LIKE 'Machine Learning%';
17
```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	first_name	last_name	course_name		
▶	Michael	Williams	Machine Learning Fundamentals		
	Daniel	Taylor	Machine Learning Fundamentals		

```

17
18 -- 3. Get all employees working in the IT department.
19 • SELECT c.first_name, c.last_name, d.department_name FROM employees e
20 JOIN citizens c ON c.id = e.citizen_id
21 JOIN departments d ON d.id = e.department_id
22 WHERE d.department_name = 'Information Technology';
23

```

Result Grid			
	Filter Rows:	Export:	Wrap Cell Content: FA
	first_name	last_name	department_name
▶	Michael	Williams	Information Technology
	Sarah	Williams	Information Technology

```

24 -- 2) Aggregate Functions
25
26 -- 4. Find total revenue generated from all orders.
27 • SELECT SUM(total_amount) AS total_revenue FROM orders
28 WHERE status = 'paid' OR status = 'delivered';
29

```

Result Grid	
	Filter Rows:
	Export:
	Wrap Cell Content: FA
	total_revenue
▶	2619.91

```

30 -- 5. Calculate average salary per department.
31 • SELECT d.id, d.department_name, AVG(s.total_salary) as average_salary FROM employees e
32 JOIN departments d ON d.id = e.department_id
33 JOIN salaries s ON s.employee_id = e.id
34 GROUP BY d.id;
35

```

Result Grid			
	Filter Rows:	Export:	Wrap Cell Content: FA
	id	department_name	average_salary
▶	1	Administration	12000.000000
	2	Information Technology	8500.000000
	3	Human Resources	11500.000000
	4	Engineering	8200.000000
	5	Marketing	10800.000000

```

36 -- 6. Retrieve top 3 highest-paid employees.
37 • SELECT CONCAT_WS(' ', c.first_name, c.last_name) as employee_name, s.total_salary
38 FROM employees e
39 JOIN citizens c ON c.id = e.citizen_id
40 JOIN salaries s ON s.employee_id = e.id
41 ORDER BY s.total_salary DESC
42 LIMIT 3;

```


Result Grid	Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
employee_name	total_salary			
Robert Johnson	12000.00			
David Thompson	11500.00			
Daniel Taylor	10800.00			


```


44 -- 3) Joins
45 -- 7. Get customer names along with their order details using INNER JOIN.
46 • SELECT c.first_name, c.last_name, o.id as order_id, o.order_date, o.status, o.total_amount
47 FROM online_shopping_customers osc
48 INNER JOIN citizens c ON c.id = osc.citizen_id
49 INNER JOIN orders o ON o.customer_id = osc.id;
50

```

Result Grid

 Filter Rows:

 Export:


 Wrap Cell Content:

	first_name	last_name	order_id	order_date	status	total_amount
▶	Robert	Johnson	1	2023-02-10 13:45:22	delivered	219.98
	David	Thompson	2	2023-02-15 09:30:15	delivered	699.99
	Jessica	Martinez	3	2023-02-20 16:20:45	delivered	89.99
	Sophia	Lee	4	2023-03-05 11:15:30	paid	1299.99
	William	Miller	5	2023-03-10 14:40:25	paid	169.98
	Jennifer	Johnson	6	2023-03-15 10:25:18	paid	139.98
	Emma	Rodriguez	7	2023-03-20 15:50:40	pending	599.99
	Emily	Thompson	8	2023-03-25 12:35:55	pending	129.98
	Matthew	Brown	9	2023-04-01 09:15:33	pending	79.99
	Andrew	Kim	10	2023-04-05 16:45:10	cancelled	219.97

```

50
51 -- 8. Retrieve doctors and their patients using LEFT JOIN.
52 • SELECT
53     CONCAT_WS(' ', cd.first_name, cd.last_name) as doctor,
54     CONCAT_WS(' ', cp.first_name, cp.last_name) as patient
55 FROM patients p
56 JOIN citizens cp ON cp.id = p.citizen_id
57 JOIN appointments a ON a.patient_id = p.id
58 JOIN doctors d ON d.id = a.doctor_id
59 JOIN citizens cd ON cd.id = d.citizen_id;
60

```





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

doctor	patient
David Thompson	Robert Johnson
David Thompson	Sophia Lee
David Thompson	Natalie Singh
James Anderson	Jennifer Johnson
James Anderson	Patricia Davis
Christopher Wilson	Jessica Martinez
Christopher Wilson	Olivia Garcia
William Miller	Michael Williams
William Miller	Matthew Brown
Richard Davis	Sarah Williams
Richard Davis	Emma Rodriguez
Andrew Kim	Daniel Taylor
Andrew Kim	Elizabeth Miller

```

60
61 -- 9. List songs and users who have added them to playlists using RIGHT JOIN
62 • SELECT
63     s.title as song_name,
64     s.artist_name,
65     CONCAT_WS(' ', c.first_name, c.last_name) as user_name,
66     p.playlist_name
67 FROM music_service_users msu
68 JOIN citizens c ON c.id = msu.citizen_id
69 JOIN playlists p ON p.user_id = msu.id
70 JOIN playlist_songs ps ON ps.playlist_id = p.id
71 RIGHT JOIN songs s ON s.id = ps.song_id;
72

```

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

song_name	artist_name	user_name	playlist_name
Shape of You	Ed Sheeran	Robert Johnson	Workout Motivation
Shape of You	Ed Sheeran	Sarah Williams	Party Mix
Shape of You	Ed Sheeran	Christopher Wilson	Morning Wake Up
Shape of You	Ed Sheeran	Matthew Brown	Weekend Vibes
Blinding Lights	The Weeknd	Sarah Williams	Party Mix
Blinding Lights	The Weeknd	Christopher Wilson	Morning Wake Up
Someone Like You	Adele	Robert Johnson	Focus Time
Someone Like You	Adele	Jessica Martinez	Chill Evening
Bad Guy	Billie Eilish	Sarah Williams	Party Mix
Bad Guy	Billie Eilish	Matthew Brown	Weekend Vibes
Uptown Funk	Mark Ronso...	Robert Johnson	Workout Motivation
Uptown Funk	Mark Ronso...	David Thompson	Road Trip Jams
Uptown Funk	Mark Ronso...	Sarah Williams	Party Mix

```

73 -- 4) Subqueries
74 -- 10. Find employees earning more than the company average salary.
75 • SELECT c.first_name, c.last_name, s.total_salary FROM employees e
76 JOIN citizens c ON c.id = e.citizen_id
77 JOIN departments d ON d.id = e.department_id
78 JOIN service_providers sp ON sp.id = d.provider_id
79 JOIN salaries s ON s.employee_id = e.id
80 WHERE s.total_salary > (
81     SELECT AVG(s.total_salary) FROM employees e
82     JOIN departments d ON d.id = e.department_id
83     JOIN service_providers sp ON sp.id = d.provider_id
84     JOIN salaries s ON s.employee_id = e.id
85     WHERE d.provider_id = 12
86 ) AND d.provider_id = 12;
87

```





Result Grid   Filter Rows: Export:  Wrap Cell Content: 

	first_name	last_name	total_salary
▶	Robert	Johnson	12000.00
	David	Thompson	11500.00
	Daniel	Taylor	10800.00

```

88 -- 11. Retrieve products that have been ordered more than 10 times.
89 • SELECT p.id, p.title, p.price, COUNT(p.id) as product_count
90 FROM products p
91 JOIN order_items oi ON oi.product_id = p.id
92 GROUP BY p.id
93 HAVING COUNT(p.id) >= 2 -- due to limited orders
94 ORDER BY product_count DESC;
95

```

Result Grid   Filter Rows: Export:  Wrap Cell Content: 

	id	title	price	product_count
▶	11	Men's Casual Shirt	39.99	2
	13	Leather Wallet	49.99	2

```

96      -- 12. Get users who have never placed an order.
97  •   SELECT c.first_name, c.last_name FROM online_shopping_customers osc
98      JOIN citizens c ON c.id = osc.citizen_id
99      LEFT JOIN orders o ON o.customer_id = osc.id
100     WHERE o.id IS NULL;
101

```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	first_name	last_name			
▶	Michael	Williams			
	Sarah	Williams			
	James	Anderson			
	Christopher	Wilson			
	Daniel	Taylor			

```

101
102      -- 5) String Functions
103      -- 13. Extract the first 3 characters of all user emails.
104  •   SELECT
105          c.id,
106          c.email,
107          LEFT(c.email, 3) AS email_prefix
108      FROM citizens c;
109

```

Result Grid				Filter Rows:	Export:	Wrap Cell Content:
	id	email	email_prefix			
▶	19	andrew.kim@email.com	and			
	8	christopher.wilson@email.com	chr			
	10	daniel.taylor@email.com	dan			
	5	david.thompson@email.com	dav			
	15	elizabeth.miller@email.com	eli			
	6	emily.thompson@email.com	emi			
	18	emma.rodriquez@email.com	emm			
	7	james.anderson@email.com	jam			
	2	jennifer.johnson@email.com	jen			
	9	jessica.martinez@email.com	jes			
	12	matthew.brown@email.com	mat			

```

110      -- 14. Convert all product names to uppercase.
111 •    SELECT
112         id,
113         UPPER(title) AS uppercase_name
114     FROM products;
115

```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	id	uppercase_name			
▶	1	SMART HOME HUB			
	2	WIRELESS EARBUDS			
	3	FITNESS TRACKER			
	4	KITCHEN BLENDER			
	5	SMART SCALE			
	6	GAMING LAPTOP			
	7	4K SMART TV			
	8	WIRELESS CHARGER			
	9	BLUETOOTH SPEAKER			
	10	DIGITAL CAMERA			
	11	MEN'S CASUAL SHIRT			
	12	WOMEN'S SUMMER D...			
	13	LEATHER WALLET			
	14	DESIGNER SUNGLAS...			
	15	ATHLETIC SHOES			

```

116      -- 15. Replace 'Inc.' with 'Ltd.' in all company names
117 •    SELECT
118         id,
119         provider_name,
120         REPLACE(provider_name, 'Inc.', 'Ltd.') AS updated_name,
121         status
122     FROM service_providers
123     WHERE provider_name LIKE '%Inc.%'
124     ORDER BY id;

```

Result Grid					Filter Rows:	Export:	Wrap Cell Content:
	id	provider_name	updated_name	status			
▶	7	TechGadgets Direct Inc.	TechGadgets Direct Ltd.	active			
	12	Smart City Corporation Inc.	Smart City Corporation Ltd.	active			