SQL JOINS

```
-- CREATE DATABASE sql_cx_live
SELECT * FROM sql_cx_live.users;
SELECT * FROM sql_cx_live.membership;
SELECT * FROM sql_cx_live.groups;
```

CROSS JOIN

```
SELECT * FROM sql_cx_live.users t1
CROSS JOIN sql_cx_live.groups t2
```

INNER JOIN

```
SELECT * FROM sql_cx_live.membership t1
INNER JOIN sql_cx_live.users t2
ON t1.user_id = t2.user_id
```

LEFT JOIN

```
SELECT * FROM sql_cx_live.membership t1
LEFT JOIN sql_cx_live.users t2
ON t1.user_id = t2.user_id
```

RIGHT JOIN

```
SELECT * FROM sql_cx_live.membership t1
RIGHT JOIN sql_cx_live.users t2
ON  t1.user_id = t2.user_id
```

SET OPERATION

UNION

```
SELECT * FROM sql_cx_live.person1
UNION
SELECT * FROM sql_cx_live.person2
```

UNION ALL

```
SELECT * FROM sql_cx_live.person1
UNION ALL
SELECT * FROM sql_cx_live.person2
```

INTERSECT

```
SELECT * FROM sql_cx_live.person1
INTERSECT
SELECT * FROM sql_cx_live.person2
```

EXCEPT

```
SELECT * FROM sql_cx_live.person1
EXCEPT
SELECT * FROM sql_cx_live.person2
```

FULL OUTER JOIN

```
SELECT * FROM sql_cx_live.membership t1
LEFT JOIN sql_cx_live.users t2
ON t1.user_id = t2.user_id
UNION
SELECT * FROM sql_cx_live.membership t1
RIGHT JOIN sql_cx_live.users t2
ON t1.user_id = t2.user_id
```

SELF JOIN

```
SELECT * FROM sql_cx_live.users t1
JOIN sql_cx_live.users t2
ON t1.emergency_contact = t2.user_id

SELECT * FROM sql_cx_live.students;
SELECT * FROM sql_cx_live.class;
```

JOIN ON MULTIPLE COLUMN

```
SELECT * FROM sql_cx_live.students t1
JOIN sql_cx_live.class t2  -- use left, right join
ON t1.class_id = t2.class_id AND t1.enrollment_year = t2.class_year
ON A NEW DATABASE
CREATE DATABASE flipkart
SELECT * FROM flipkart.users;
SELECT * FROM flipkart.orders;
SELECT * FROM flipkart.order_details;
SELECT * FROM flipkart.category;
```

JOIN TWO TABLES TOGETHER

```
SELECT * FROM flipkart.order_details t1
JOIN flipkart.orders t2
ON t1.order_id = t2.order_id
JOIN flipkart.users t3
ON t2.user_id = t3.user_id
```

```
-- Filtering column from triple join tables
SELECT t1.order_id, t1.amount, t1.profit, t3.name
FROM flipkart.order_details t1
JOIN flipkart.orders t2
ON  t1.order_id = t2.order_id
JOIN flipkart.users t3
ON t2.user_id = t3.user_id
```

```
-- Find order_id, name, and city by joining users and orders

SELECT t2.order_id, t1.name, t1.city

FROM flipkart.users t1

JOIN flipkart.orders t2

ON t1.user_id = t2.user_id



-- Find order_id, product category by joining order_details and category

SELECT t1.order_id, t2.vertical

FROM flipkart.order_details t1

JOIN flipkart.category t2

ON t1.category_id = t2.category_id
```

```
-- Find orders come from pune city
SELECT t1.user_id, t1.name,t3.amount, t4.vertical
FROM flipkart.users t1
JOIN flipkart.orders t2
ON t1.user_id = t2.user_id
JOIN flipkart.order_details t3
ON t2.order_id = t3.order_id
JOIN flipkart.category t4
ON t3.category_id = t4.category_id
WHERE city = 'pune'

SELECT * FROM flipkart.orders t1
JOIN flipkart.users t2
ON t1.user_id = t2.user_id
WHERE t2.city = 'pune' AND t2.name = 'Sarita'
```

```
-- Find all profitable orders
SELECT t2.vertical, SUM(profit) AS 'total_profit'
FROM flipkart.order_details t1
JOIN flipkart.category t2
ON t1.category_id = t2.category_id
WHERE profit>0
GROUP BY vertical

-- Find the customer who has placed max number of order
SELECT name, COUNT(*) AS 'total_order' FROM flipkart.users t1
JOIN flipkart.orders  t2
ON t1.user_id = t2.user_id
GROUP BY name
ORDER BY total_order DESC
```

```
-- Find all categories with profit higher than 5000
SELECT t1.vertical, SUM(t2.profit) AS 'Profit' FROM
flipkart.category t1
JOIN flipkart.order_details t2
ON t1.category_id = t2.category_id
WHERE profit>0
GROUP BY vertical
HAVING Profit > 5000

-- Which is the most profitable category
-- which is the most profitable state
```