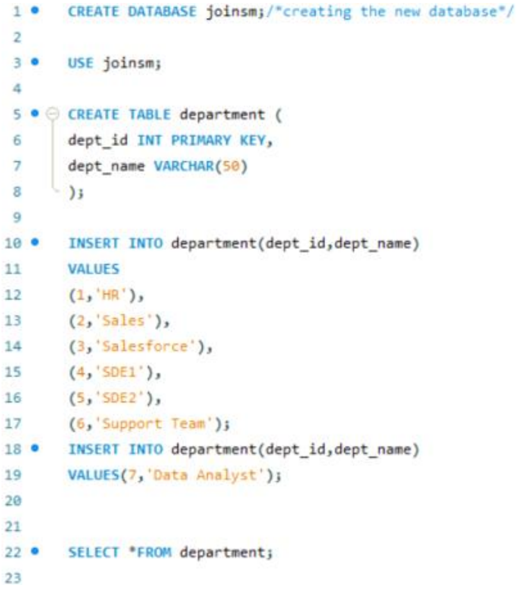
JOINS & PRACTICE

13 February 2025 13:00

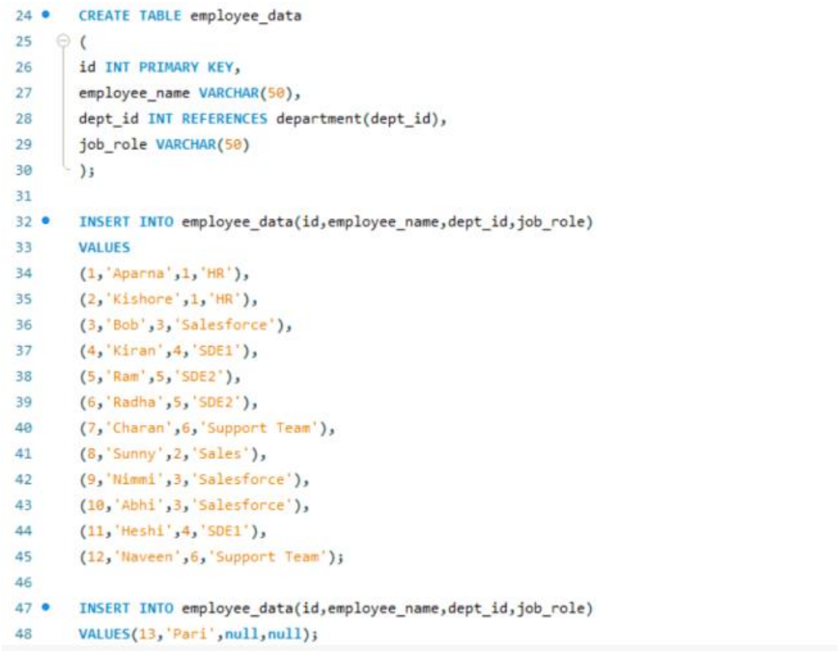
Joins in DBMS

Joins combine data from two or more tables based on a related column.

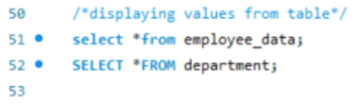
- **INNER JOIN** → Returns only matching records from both tables.
- **LEFT JOIN** (LEFT OUTER) → Returns all records from the left table + matching records from the right.
- **RIGHT JOIN** (RIGHT OUTER) → Returns all records from the right table + matching records from the left.
- **FULL JOIN** (FULL OUTER) → Returns all records from both tables, with NULLs where no match exists.
- **CROSS JOIN** → Returns the Cartesian product (every row in table A with every row in table B).
- **SELF JOIN** → Joins a table with itself.



```sql
CREATE DATABASE joinsm;/*creating the new database*/

USE joinsm;

CREATE TABLE department (
dept_id INT PRIMARY KEY,
dept_name VARCHAR(50)
);

INSERT INTO department(dept_id,dept_name)
VALUES
(1,'HR'),
(2,'Sales'),
(3,'Salesforce'),
(4,'SDE1'),
(5,'SDE2'),
(6,'Support Team');
INSERT INTO department(dept_id,dept_name)
VALUES(7,'Data Analyst');


SELECT *FROM department;

```

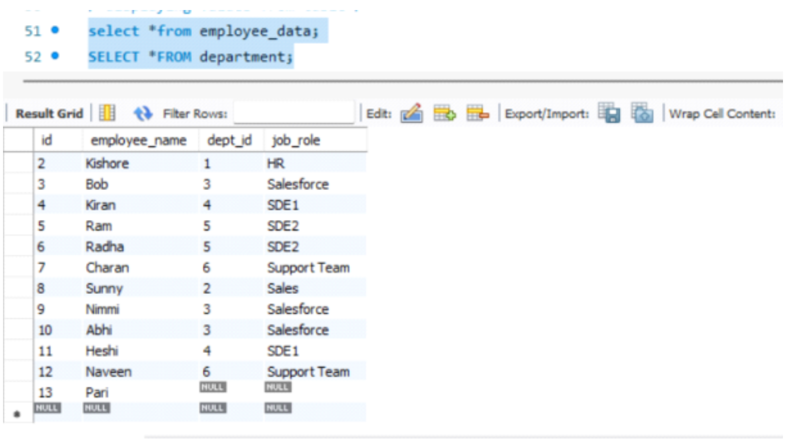


```sql
CREATE TABLE employee_data
(
id INT PRIMARY KEY,
employee_name VARCHAR(50),
dept_id INT REFERENCES department(dept_id),
job_role VARCHAR(50)
);

INSERT INTO employee_data(id,employee_name,dept_id,job_role)
VALUES
(1,'Aparna',1,'HR'),
(2,'Kishore',1,'HR'),
(3,'Bob',3,'Salesforce'),
(4,'Kiran',4,'SDE1'),
(5,'Ram',5,'SDE2'),
(6,'Radha',5,'SDE2'),
(7,'Charan',6,'Support Team'),
(8,'Sunny',2,'Sales'),
(9,'Nimmi',3,'Salesforce'),
(10,'Abhi',3,'Salesforce'),
(11,'Heshi',4,'SDE1'),
(12,'Naveen',6,'Support Team');

INSERT INTO employee_data(id,employee_name,dept_id,job_role)
VALUES(13,'Pari',null,null);
```



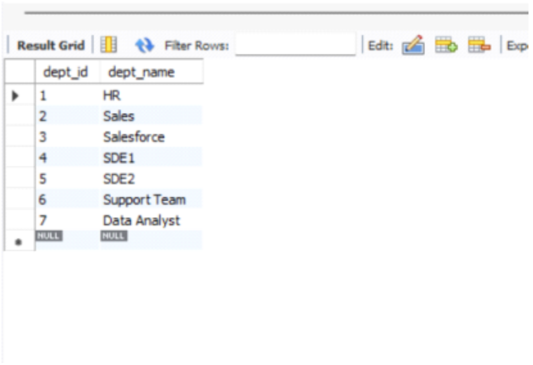
```sql
/*displaying values from table*/
select *from employee_data;
SELECT *FROM department;

```

Output so far

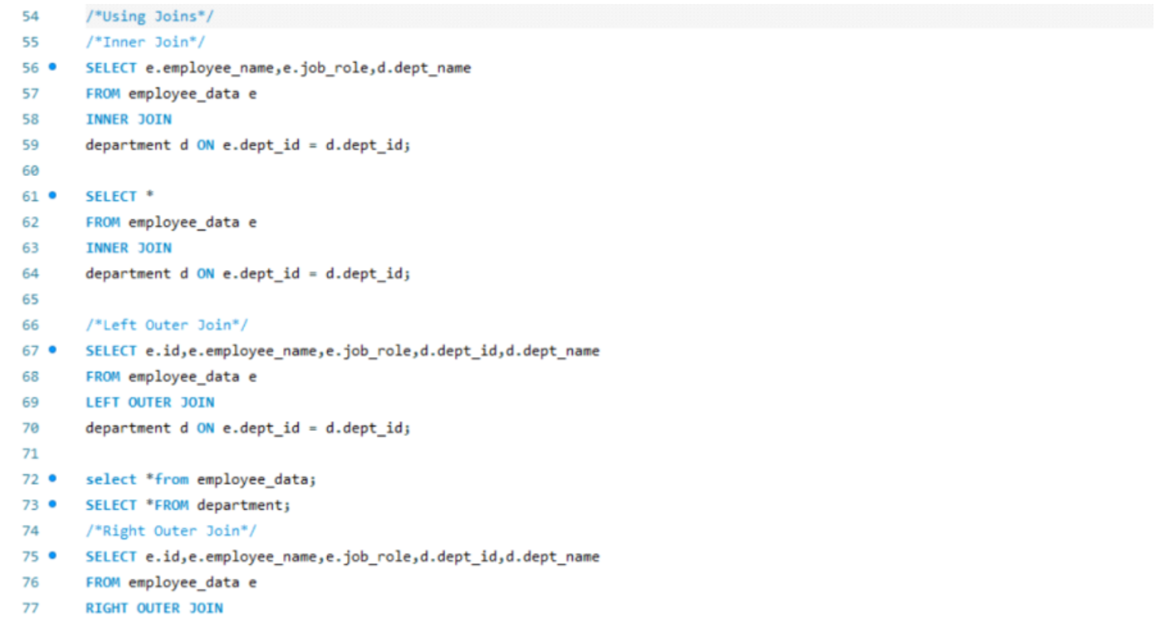


```
select *from employee_data;
SELECT *FROM department;
```

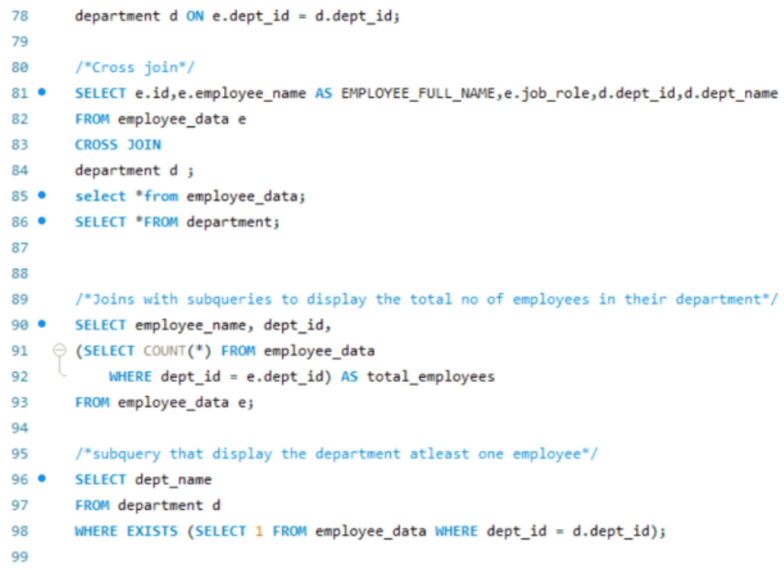
id	employee_name	dept_id	job_role
2	Kishore	1	HR
3	Bob	3	Salesforce
4	Kiran	4	SDE1
5	Ram	5	SDE2
6	Radha	5	SDE2
7	Charan	6	Support Team
8	Sunny	2	Sales
9	Nimmi	3	Salesforce
10	Abhi	3	Salesforce
11	Heshi	4	SDE1
12	Naveen	6	Support Team
13	Pari	NULL	NULL
NULL	NULL	NULL	NULL



dept_id	dept_name
1	HR
2	Sales
3	Salesforce
4	SDE1
5	SDE2
6	Support Team
7	Data Analyst
NULL	NULL



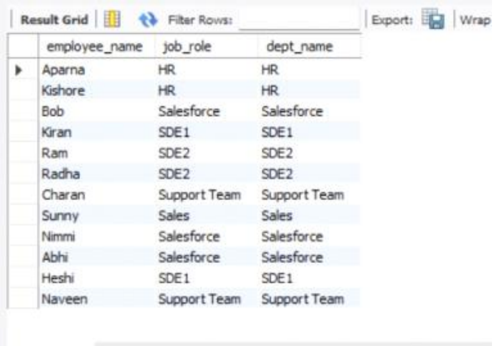
```
/*Using Joins*/
/*Inner Join*/
SELECT e.employee_name,e.job_role,d.dept_name
FROM employee_data e
INNER JOIN
department d ON e.dept_id = d.dept_id;

SELECT *
FROM employee_data e
INNER JOIN
department d ON e.dept_id = d.dept_id;

/*Left Outer Join*/
SELECT e.id,e.employee_name,e.job_role,d.dept_id,d.dept_name
FROM employee_data e
LEFT OUTER JOIN
department d ON e.dept_id = d.dept_id;

select *from employee_data;
SELECT *FROM department;
/*Right Outer Join*/
SELECT e.id,e.employee_name,e.job_role,d.dept_id,d.dept_name
FROM employee_data e
RIGHT OUTER JOIN
```



```
department d ON e.dept_id = d.dept_id;

/*Cross join*/
SELECT e.id,e.employee_name AS EMPLOYEE_FULL_NAME,e.job_role,d.dept_id,d.dept_name
FROM employee_data e
CROSS JOIN
department d ;
select *from employee_data;
SELECT *FROM department;


/*Joins with subqueries to display the total no of employees in their department*/
SELECT employee_name, dept_id,
(SELECT COUNT(*) FROM employee_data
    WHERE dept_id = e.dept_id) AS total_employees
FROM employee_data e;

/*subquery that display the department atleast one employee*/
SELECT dept_name
FROM department d
WHERE EXISTS (SELECT 1 FROM employee_data WHERE dept_id = d.dept_id);

```

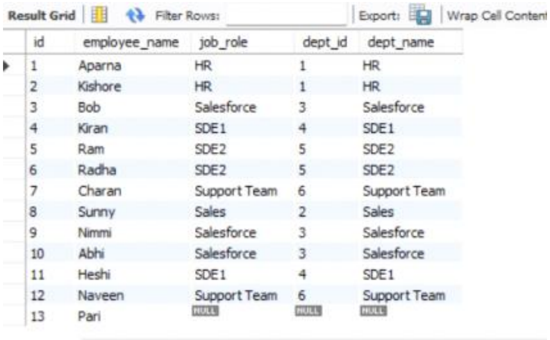
Outputs

1) Inner join



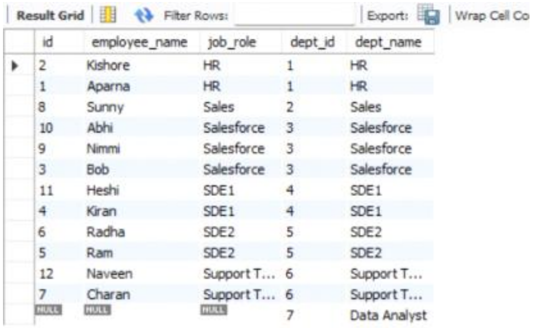
employee_name	job_role	dept_name
Aparna	HR	HR
Kishore	HR	HR
Bob	Salesforce	Salesforce
Kiran	SDE1	SDE1
Ram	SDE2	SDE2
Radha	SDE2	SDE2
Charan	Support Team	Support Team
Sunny	Sales	Sales
Nimmi	Salesforce	Salesforce
Abhi	Salesforce	Salesforce
Heshi	SDE1	SDE1
Naveen	Support Team	Support Team

2) Left outer join



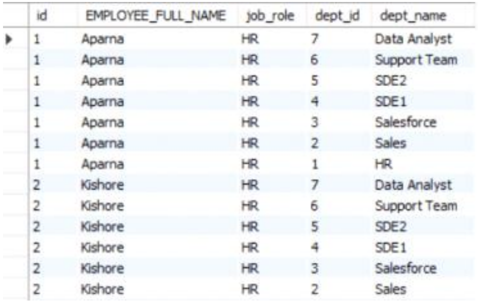
id	employee_name	job_role	dept_id	dept_name
1	Aparna	HR	1	HR
2	Kishore	HR	1	HR
3	Bob	Salesforce	3	Salesforce
4	Kiran	SDE1	4	SDE1
5	Ram	SDE2	5	SDE2
6	Radha	SDE2	5	SDE2
7	Charan	Support Team	6	Support Team
8	Sunny	Sales	2	Sales
9	Nimmi	Salesforce	3	Salesforce
10	Abhi	Salesforce	3	Salesforce
11	Heshi	SDE1	4	SDE1
12	Naveen	Support Team	6	Support Team
13	Pari	NULL	NULL	NULL

3) Right outer join



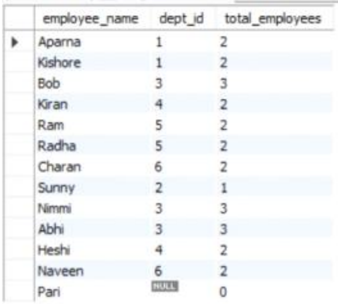
id	employee_name	job_role	dept_id	dept_name
2	Kishore	HR	1	HR
1	Aparna	HR	1	HR
8	Sunny	Sales	2	Sales
10	Abhi	Salesforce	3	Salesforce
9	Nimmi	Salesforce	3	Salesforce
3	Bob	Salesforce	3	Salesforce
11	Heshi	SDE1	4	SDE1
4	Kiran	SDE1	4	SDE1
6	Radha	SDE2	5	SDE2
5	Ram	SDE2	5	SDE2
12	Naveen	Support T...	6	Support T...
7	Charan	Support T...	6	Support T...
NULL	NULL	NULL	7	Data Analyst

4) Cross join



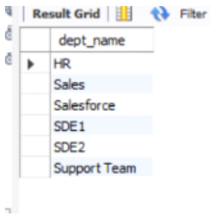
id	EMPLOYEE_FULL_NAME	job_role	dept_id	dept_name
1	Aparna	HR	7	Data Analyst
1	Aparna	HR	6	Support Team
1	Aparna	HR	5	SDE2
1	Aparna	HR	4	SDE1
1	Aparna	HR	3	Salesforce
1	Aparna	HR	2	Sales
1	Aparna	HR	1	HR
2	Kishore	HR	7	Data Analyst
2	Kishore	HR	6	Support Team
2	Kishore	HR	5	SDE2
2	Kishore	HR	4	SDE1
2	Kishore	HR	3	Salesforce
2	Kishore	HR	2	Sales

5) Joins with subqueries to display the total no of employees in their department



employee_name	dept_id	total_employees
Aparna	1	2
Kishore	1	2
Bob	3	3
Kiran	4	2
Ram	5	2
Radha	5	2
Charan	6	2
Sunny	2	1
Nimmi	3	3
Abhi	3	3
Heshi	4	2
Naveen	6	2
Pari	NULL	0

6) subquery that display the department atleast one employee



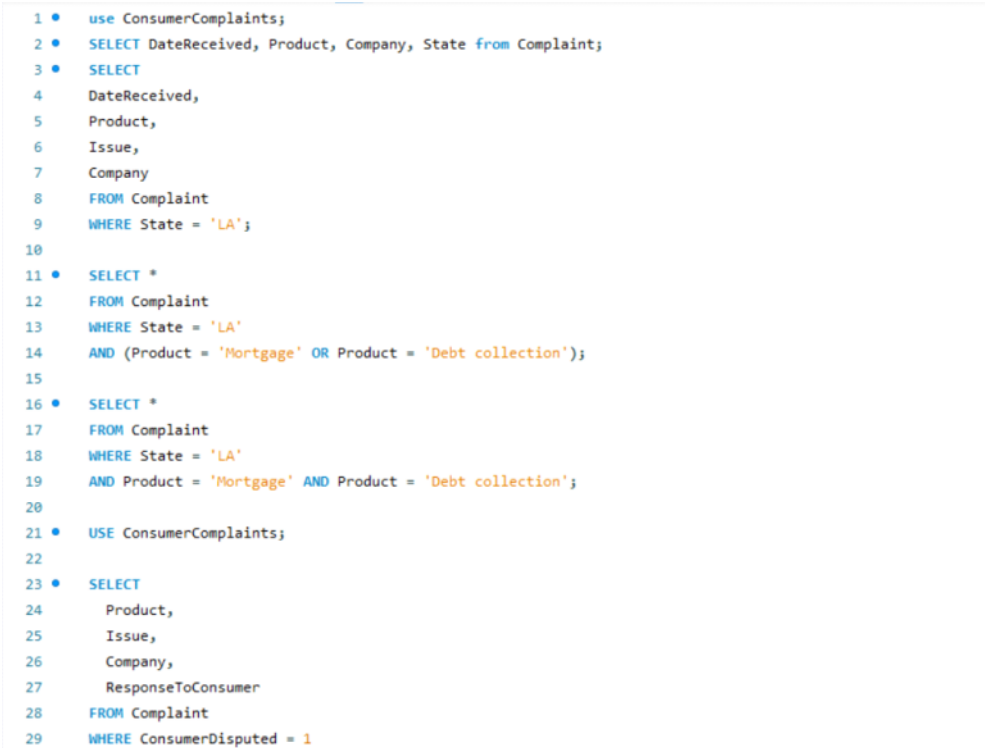
dept_name
HR
Sales
Salesforce
SDE1
SDE2
Support Team

Sorting, Grouping & Aggregates in DBMS

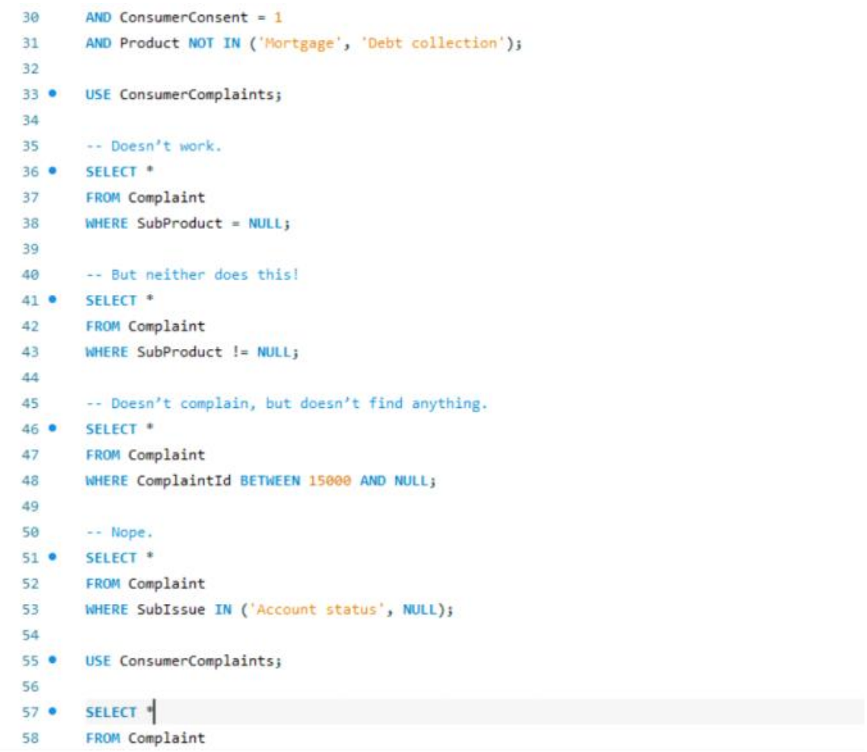
- **Sorting (ORDER BY)** → Arranges records in ascending (ASC) or descending (DESC) order.
  - Example: SELECT * FROM Students ORDER BY Age DESC;
- **Grouping (GROUP BY)** → Groups rows with the same values in a column. Used with aggregate functions.
  - Example: SELECT Department, COUNT(*) FROM Employees GROUP BY Department;
- **Aggregate Functions** → Perform calculations on groups of rows.
  - COUNT() → Counts rows
  - SUM() → Adds values
  - AVG() → Calculates average
  - MAX() → Finds max value
  - MIN() → Finds min value

SQL Operators & Constraints

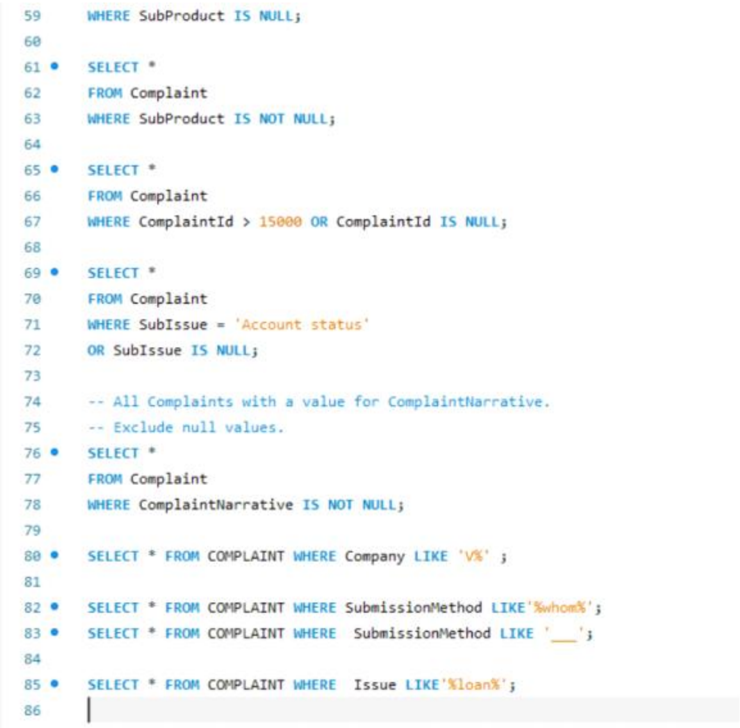
- **LIKE** → Pattern matching using wildcards (% for multiple chars, _ for a single char).
  - Example: SELECT * FROM Users WHERE Name LIKE 'A%'; (Names starting with 'A')
- **NOT** → Negates a condition.
  - Example: SELECT * FROM Students WHERE Age NOT BETWEEN 18 AND 25;
- **NULL & NOT NULL** → Checks for missing or non-missing values.
  - Example: SELECT * FROM Orders WHERE Discount IS NOT NULL;
- **IN** → Matches any value in a given list.
  - Example: SELECT * FROM Employees WHERE Department IN ('HR', 'IT', 'Sales');
- **BETWEEN** → Filters values within a range (inclusive).
  - Example: SELECT * FROM Products WHERE Price BETWEEN 100 AND 500;



```sql
use ConsumerComplaints;
SELECT DateReceived, Product, Company, State from Complaint;
SELECT
DateReceived,
Product,
Issue,
Company
FROM Complaint
WHERE State = 'LA';

SELECT *
FROM Complaint
WHERE State = 'LA'
AND (Product = 'Mortgage' OR Product = 'Debt collection');

SELECT *
FROM Complaint
WHERE State = 'LA'
AND Product = 'Mortgage' AND Product = 'Debt collection';

USE ConsumerComplaints;

SELECT
  Product,
  Issue,
  Company,
  ResponseToConsumer
FROM Complaint
WHERE ConsumerDisputed = 1
```

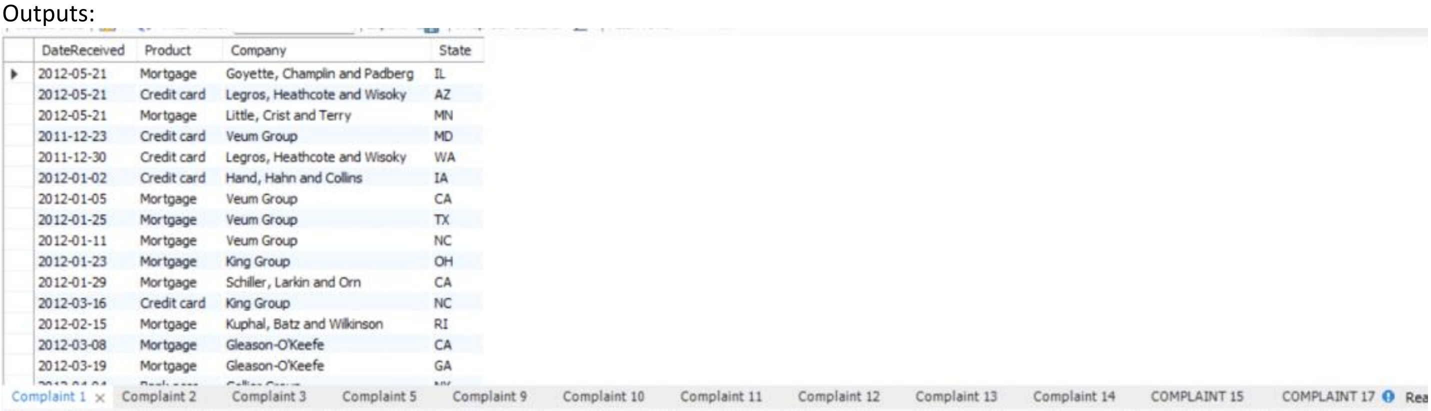


```
AND ConsumerConsent = 1
AND Product NOT IN ('Mortgage', 'Debt collection');

USE ConsumerComplaints;

-- Doesn't work.
SELECT *
FROM Complaint
WHERE SubProduct = NULL;

-- But neither does this!
SELECT *
FROM Complaint
WHERE SubProduct != NULL;

-- Doesn't complain, but doesn't find anything.
SELECT *
FROM Complaint
WHERE ComplaintId BETWEEN 15000 AND NULL;

-- Nope.
SELECT *
FROM Complaint
WHERE SubIssue IN ('Account status', NULL);

USE ConsumerComplaints;

SELECT *
FROM Complaint
```

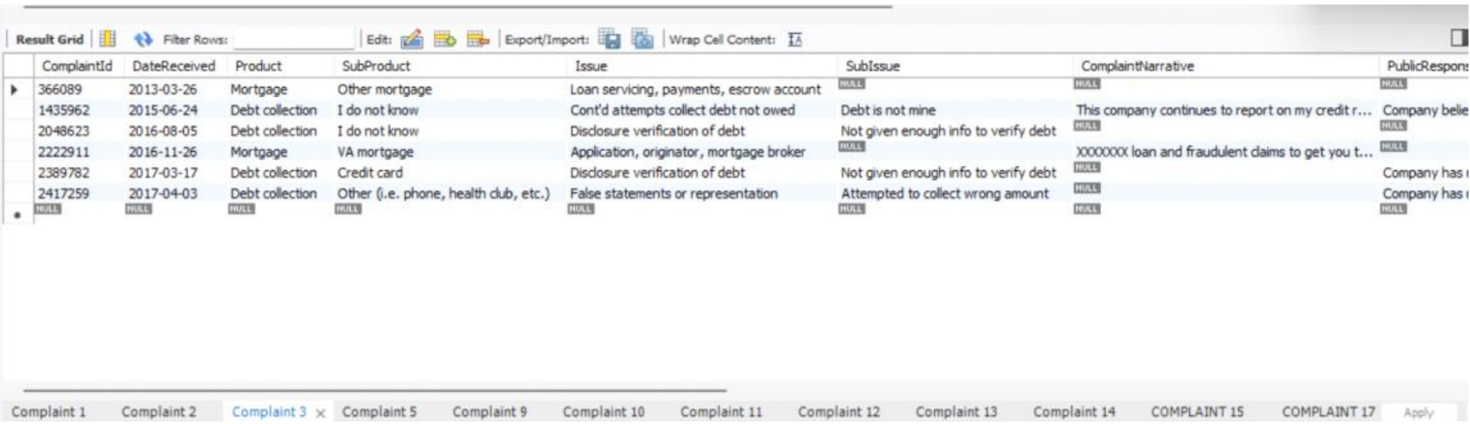


```
WHERE SubProduct IS NULL;

SELECT *
FROM Complaint
WHERE SubProduct IS NOT NULL;

SELECT *
FROM Complaint
WHERE ComplaintId > 15000 OR ComplaintId IS NULL;

SELECT *
FROM Complaint
WHERE SubIssue = 'Account status'
OR SubIssue IS NULL;

-- All Complaints with a value for ComplaintNarrative.
-- Exclude null values.
SELECT *
FROM Complaint
WHERE ComplaintNarrative IS NOT NULL;

SELECT * FROM COMPLAINT WHERE Company LIKE 'V%' ;

SELECT * FROM COMPLAINT WHERE SubmissionMethod LIKE'%whom%';
SELECT * FROM COMPLAINT WHERE  SubmissionMethod LIKE '___';

SELECT * FROM COMPLAINT WHERE  Issue LIKE'%loan%';
```

Outputs:



DateReceived	Product	Company	State
2012-05-21	Mortgage	Goyette, Champlin and Padberg	IL
2012-05-21	Credit card	Legros, Heathcote and Wisoky	AZ
2012-05-21	Mortgage	Little, Crist and Terry	MN
2011-12-23	Credit card	Veum Group	MD
2011-12-30	Credit card	Legros, Heathcote and Wisoky	WA
2012-01-02	Credit card	Hand, Hahn and Collins	IA
2012-01-05	Mortgage	Veum Group	CA
2012-01-25	Mortgage	Veum Group	TX
2012-01-11	Mortgage	Veum Group	NC
2012-01-23	Mortgage	King Group	OH
2012-01-29	Mortgage	Schiller, Larkin and Orn	CA
2012-03-16	Credit card	King Group	NC
2012-02-15	Mortgage	Kuphal, Batz and Wilkinson	RI
2012-03-08	Mortgage	Gleason-O'Keefe	CA
2012-03-19	Mortgage	Gleason-O'Keefe	GA



ComplaintId	DateReceived	Product	SubProduct	Issue	SubIssue	ComplaintNarrative	PublicRespons
366089	2013-03-26	Mortgage	Other mortgage	Loan servicing, payments, escrow account	NULL	NULL	NULL
1435962	2015-06-24	Debt collection	I do not know	Cont'd attempts collect debt not owed	Debt is not mine	This company continues to report on my credit r...	Company belie
2048623	2016-08-05	Debt collection	I do not know	Disclosure verification of debt	Not given enough info to verify debt	NULL	NULL
2222911	2016-11-26	Mortgage	VA mortgage	Application, originator, mortgage broker	NULL	XXXXXXX loan and fraudulent claims to get you t...	NULL
2389782	2017-03-17	Debt collection	Credit card	Disclosure verification of debt	Not given enough info to verify debt	NULL	Company has
2417259	2017-04-03	Debt collection	Other (i.e. phone, health club, etc.)	False statements or representation	Attempted to collect wrong amount	NULL	Company has
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL