

TASK-1 : Basic Data Retrieval & Filtering - Complete Breakdown (OUTPUTS)

1. Column Selection :

- Retrieve specific columns instead of entire tables (SELECT col1, col2 vs SELECT *)

Query Query History

```
1 ✓ SELECT first_name, last_name, department
2   FROM employees;
3   -- avoiding SELECT *
4
```

Data Output Messages Notifications

first_name character varying (50) 🔒 last_name character varying (50) 🔒 department character varying (50) 🔒

	first_name	last_name	department
1	John	Smith	Sales
2	Sarah	Stone	Sales
3	Amit	Shah	Marketing
4	Neha	Singh	Marketing
5	Ravi	Patel	IT
6	Kiran	Saxena	Sales
7	Anil	Verma	HR
8	Pooja	Sharma	IT

- Handle calculated columns (e.g., `SELECT salary * 1.1 AS new_salary`)

Query Query History

```
1 v SELECT
2     first_name,
3     salary,
4     salary * 1.10 AS increased_salary
5 FROM employees;
6
```

Data Output Messages Notifications



	first_name character varying (50) 	salary numeric (10,2) 	increased_salary numeric 
1	John	45000.00	49500.0000
2	Sarah	42000.00	46200.0000
3	Amit	60000.00	66000.0000
4	Neha	52000.00	57200.0000
5	Ravi	75000.00	82500.0000
6	Kiran	48000.00	52800.0000
7	Anil	40000.00	44000.0000
8	Pooja	82000.00	90200.0000

2. Precision Filtering

- Apply WHERE clauses with:
 - Equality operators (`=`, `!=`)

Query Query History

```
1 v SELECT first_name, department
2 FROM employees
3 WHERE department = 'Sales';
```

Data Output Messages Notifications



	first_name character varying (50) 	department character varying (50) 
1	John	Sales
2	Sarah	Sales
3	Kiran	Sales

Query Query History

```
1 ▾ SELECT first_name, gender
2   FROM employees
3   WHERE gender != 'M';
```

Data Output Messages Notifications



	first_name character varying (50)	gender character (1)
1	Sarah	F
2	Neha	F
3	Kiran	F
4	Pooja	F

- o Numerical comparisons (>, <, >=, <=)

Query Query History

```
1 ▾ SELECT first_name, salary
2   FROM employees
3   WHERE salary > 50000;
4
```

Data Output Messages Notifications



	first_name character varying (50)	salary numeric (10,2)
1	Amit	60000.00
2	Neha	52000.00
3	Ravi	75000.00
4	Pooja	82000.00

Query Query History

```
1 ▾ SELECT first_name, hire_date
2   FROM employees
3 WHERE hire_date >= '2023-01-01';
4 |
```

Data Output Messages Notifications

A screenshot of a database query tool interface. At the top, there are tabs for "Query" and "Query History", with "Query" being the active tab. Below the tabs is a code editor containing a SQL SELECT statement. The statement selects the columns "first_name" and "hire_date" from the table "employees". The condition in the WHERE clause is "hire_date >= '2023-01-01'". There are four numbered lines in the code editor, with the fourth line being a blank line. Below the code editor is a toolbar with various icons. Underneath the toolbar is a table with two rows of data. The table has two columns: "first_name" and "hire_date". The first row contains the values "Amit" and "2023-01-10". The second row contains the values "Neha" and "2023-06-05". Both columns have a lock icon next to their names.

	first_name character varying (50)	hire_date date
1	Amit	2023-01-10
2	Neha	2023-06-05

o Range operators (BETWEEN for numbers/dates)

Query Query History

```
1 ▾ SELECT first_name, hire_date
2   FROM employees
3 WHERE hire_date BETWEEN '1995-01-01' AND '1999-12-31';
4 |
```

Data Output Messages Notifications

A screenshot of a database query tool interface, similar to the one above. The "Query" tab is active. The code editor contains a SQL SELECT statement with a WHERE clause using the BETWEEN operator. The statement selects "first_name" and "hire_date" from the "employees" table, filtering for hire dates between "1995-01-01" and "1999-12-31". There are four numbered lines in the code editor. Below the code editor is a toolbar with icons. Underneath the toolbar is a table with three rows of data. The table has two columns: "first_name" and "hire_date". The first row contains "John" and "1995-03-15". The second row contains "Sarah" and "1998-07-20". The third row contains "Kiran" and "1999-12-24". Both columns have a lock icon next to their names.

	first_name character varying (50)	hire_date date
1	John	1995-03-15
2	Sarah	1998-07-20
3	Kiran	1999-12-24

o Text pattern matching (LIKE, NOT LIKE with % and _ wildcards)

Query Query History

```
1 ▾ SELECT first_name, last_name
2   FROM employees
3   WHERE last_name LIKE 'S%';
4
```

Data Output Messages Notifications

The screenshot shows a database interface with a query editor at the top containing a SQL SELECT statement. Below the editor is a toolbar with various icons. The main area displays a table with six rows of data, representing the results of the query. The columns are labeled 'first_name' and 'last_name'. The data consists of six entries: 1. John Smith, 2. Sarah Stone, 3. Amit Shah, 4. Neha Singh, 5. Kiran Saxena, and 6. Pooja Sharma.

	first_name character varying (50)	last_name character varying (50)
1	John	Smith
2	Sarah	Stone
3	Amit	Shah
4	Neha	Singh
5	Kiran	Saxena
6	Pooja	Sharma

Query Query History

```
1 ▾ SELECT first_name, email
2   FROM employees
3   WHERE email NOT LIKE '%gmail%';
4
```

Data Output Messages Notifications



	first_name character varying (50) 	email character varying (100) 
1	John	john.smith@company.com
2	Sarah	sarah.stone@company.com
3	Amit	amit.shah@company.com
4	Neha	neha.singh@company.com
5	Ravi	ravi.patel@company.com
6	Kiran	kiran.saxena@company.com
7	Anil	anil.verma@company.com
8	Pooja	pooja.sharma@company.com

- Combine conditions with AND/OR

Query Query History

```
1 ▾ SELECT first_name, department, salary
2   FROM employees
3   WHERE department = 'Sales'
4   AND (salary > 45000 OR commission IS NOT NULL);
5
```

Data Output Messages Notifications



	first_name character varying (50) 	department character varying (50) 	salary numeric (10,2) 
1	Sarah	Sales	42000.00
2	Kiran	Sales	48000.00

- Handle NULL values (IS NULL/IS NOT NULL)

Query Query History

```
1 ✓ SELECT first_name
2   FROM employees
3 WHERE manager_id IS NULL;
4
```

Data Output Messages Notifications



	first_name character varying (50) 
1	John
2	Amit
3	Ravi
4	Anil

3. Data Organization:

- Sort results using ORDER BY (single/multiple columns)

Query Query History

```
1 ✓ SELECT first_name, last_name, department
2   FROM employees;
3   -- avoiding SELECT *
4
```

Data Output Messages Notifications



	first_name character varying (50) 	last_name character varying (50) 	department character varying (50) 
1	John	Smith	Sales
2	Sarah	Stone	Sales
3	Amit	Shah	Marketing
4	Neha	Singh	Marketing
5	Ravi	Patel	IT
6	Kiran	Saxena	Sales
7	Anil	Verma	HR
8	Pooja	Sharma	IT

- Control sort direction (ASC for ascending, DESC for descending)

Query Query History

```

1 v SELECT first_name, last_name, department
2   FROM employees;
3   -- avoiding SELECT *
4

```

Data Output Messages Notifications

SQL

	first_name character varying (50)	last_name character varying (50)	department character varying (50)
1	John	Smith	Sales
2	Sarah	Stone	Sales
3	Amit	Shah	Marketing
4	Neha	Singh	Marketing
5	Ravi	Patel	IT
6	Kiran	Saxena	Sales
7	Anil	Verma	HR
8	Pooja	Sharma	IT

- Implement combined sorts (e.g., ORDER BY department ASC, salary DESC)

Query Query History

```

1 v SELECT first_name, department, salary
2   FROM employees
3   ORDER BY department ASC, salary DESC;
4

```

Data Output Messages Notifications

SQL

	first_name character varying (50)	department character varying (50)	salary numeric (10,2)
1	Anil	HR	40000.00
2	Pooja	IT	82000.00
3	Ravi	IT	75000.00
4	Amit	Marketing	60000.00
5	Neha	Marketing	52000.00
6	Kiran	Sales	48000.00
7	John	Sales	45000.00
8	Sarah	Sales	42000.00

4. Output Control:

- Limit results with LIMIT (MySQL/PostgreSQL) or TOP (SQL Server)

Query Query History

```
1 ▾ SELECT first_name
2   FROM employees
3   LIMIT 5;
```

Data Output Messages Notifications

first_name
character varying (50)

	first_name
1	John
2	Sarah
3	Amit
4	Neha
5	Ravi

- Paginate results using OFFSET

Query Query History

```
1 ▾ SELECT first_name
2   FROM employees
3   LIMIT 2 OFFSET 2;
4
```

Data Output Messages Notifications

first_name
character varying (50)

	first_name
1	Amit
2	Neha

5. Data Validation:

- Verify query accuracy through record counts and sample checks

Query Query History

```
1 ▾ SELECT COUNT(*)
2   FROM employees
3   WHERE department = 'Marketing';
4
```

Data Output Messages Notifications

≡+

	count	bigint
1	2	

- Compare output against source data integrity

Query Query History

```
1 ▾ SELECT
2   MIN(salary),
3   MAX(salary),
4   MIN(hire_date),
5   MAX(hire_date)
6   FROM employees;
7
```

Data Output Messages Notifications

≡+

	min	max	min	max
	numeric	numeric	date	date
1	40000.00	82000.00	1995-03-15	2023-06-05

