

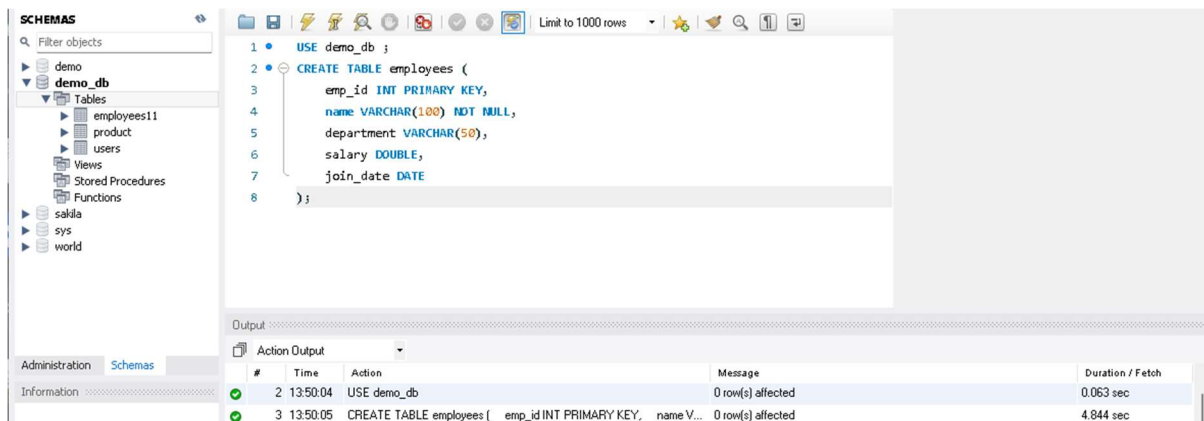
SQL Assignment

Name: Arya Dilliwale

AF04954002

1) CREATING TABLE

```
CREATE TABLE employees (  
    emp_id INT PRIMARY KEY,  
    name VARCHAR(100) NOT NULL,  
    department VARCHAR(50),  
    salary DOUBLE,  
    join_date DATE
```



2) INSERT QUERY

```
INSERT INTO employees (emp_id, name, department, salary, join_date) VALUES  
(101, 'John Doe', 'HR', 45000, '2021-06-15'),  
(102, 'Jane Smith', 'IT', 75000, '2020-01-10'),  
(103, 'Alice Johnson', 'Finance', 60000, '2019-08-23'),  
(104, 'Bob Brown', 'IT', 80000, '2022-03-01'),
```

(105, 'Eve Davis', 'Marketing', 55000, '2021-11-05');

The screenshot shows a SQL Studio interface. On the left, the 'SCHEMAS' pane displays a tree view with 'demo' and 'demo_db' (containing tables like employees11, product, users, Views, Stored Procedures, Functions, sakila, sys, and world). The main editor contains a SQL script with 7 lines: 1. USE demo_db ; 2. INSERT INTO employees (emp_id, name, department, salary, join_date) VALUES 3. (101, 'John Doe', 'HR', 45000, '2021-06-15'), 4. (102, 'Jane Smith', 'IT', 75000, '2020-01-10'), 5. (103, 'Alice Johnson', 'Finance', 60000, '2019-08-23'), 6. (104, 'Bob Brown', 'IT', 80000, '2022-03-01'), 7. (105, 'Eve Davis', 'Marketing', 55000, '2021-11-05');. The 'Output' pane at the bottom shows the execution results for the INSERT statement, indicating 5 rows affected.

#	Time	Action	Message	Duration / Fetch
4	13:51:17	USE demo_db	0 row(s) affected	0.000 sec
5	13:51:17	INSERT INTO employees (emp_id, name, department, salary, join_date) ...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.968 sec

3) SELECT QUERY

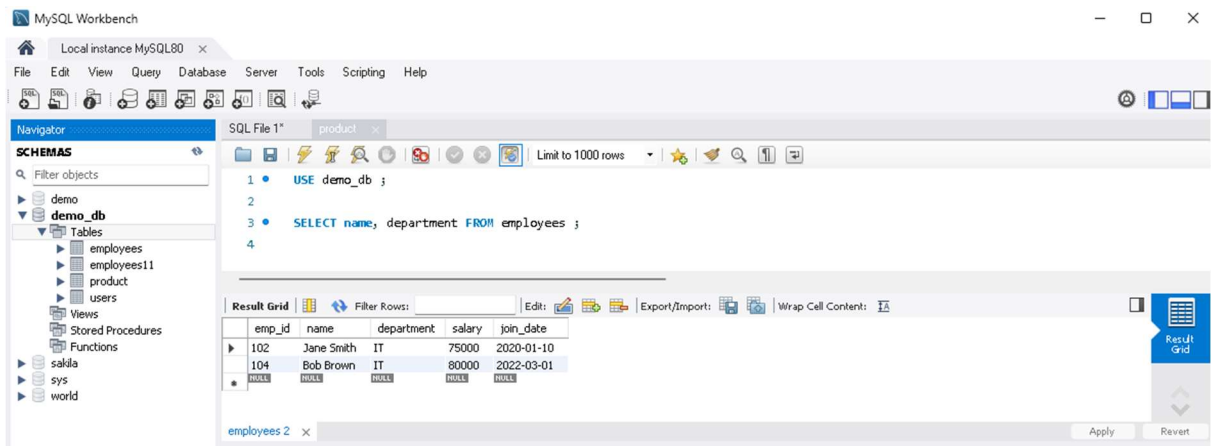
- SELECT * FROM employees;

The screenshot shows the same SQL Studio interface. The main editor now contains a SQL script with 3 lines: 1. USE demo_db ; 2. 3. SELECT * FROM employees ;. The 'Result Grid' pane displays the results of the SELECT query, showing 5 rows of employee data. The 'Output' pane at the bottom shows the execution results for the SELECT statement, indicating 5 rows returned.

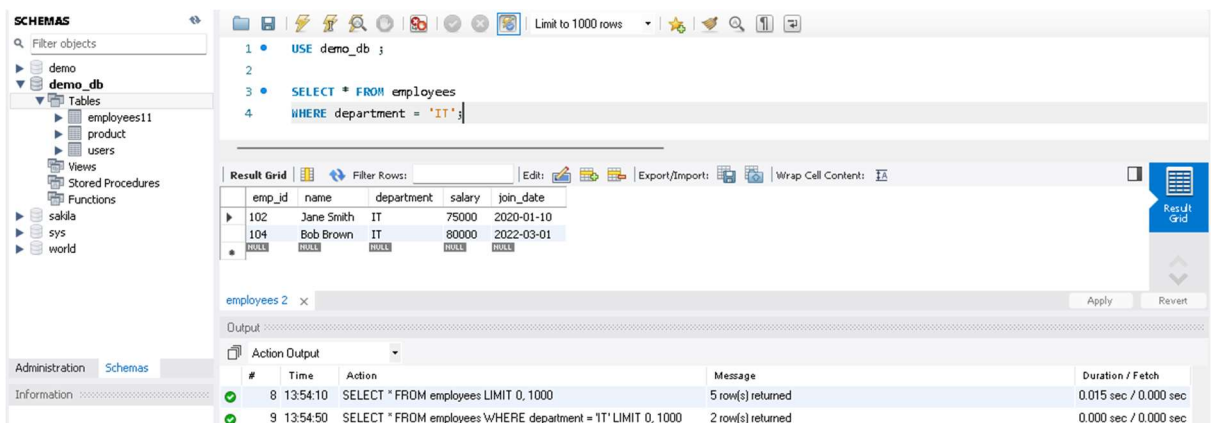
emp_id	name	department	salary	join_date
101	John Doe	HR	45000	2021-06-15
102	Jane Smith	IT	75000	2020-01-10
103	Alice Johnson	Finance	60000	2019-08-23
104	Bob Brown	IT	80000	2022-03-01
105	Eve Davis	Marketing	55000	2021-11-05

#	Time	Action	Message	Duration / Fetch
6	13:53:55	SELECT * FROM employees LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

- SELECT name, department FROM employees;

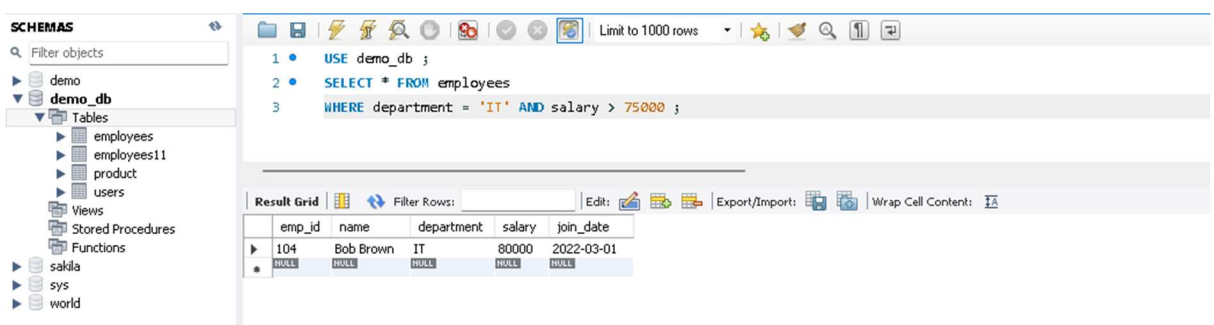


- `SELECT * FROM employees WHERE department = 'IT';`



4) AND, IN BETWEEN & LIKE

- `SELECT * FROM employees WHERE department = 'IT' AND salary > 75000;`



- `SELECT * FROM employees WHERE department IN ('IT', 'Finance');`

The screenshot shows a SQL IDE interface. On the left, the 'SCHEMAS' pane displays a tree view with 'demo_db' expanded, showing tables like 'employees', 'employees11', 'product', 'users', 'Views', 'Stored Procedures', 'Functions', 'sakila', 'sys', and 'world'. The main query editor contains the following SQL code:

```
1 • USE demo_db ;
2 • SELECT * FROM employees
3 • WHERE department IN ('IT','Finance');
```

The 'Result Grid' at the bottom shows the results of the query:

emp_id	name	department	salary	join_date
104	Bob Brown	IT	80000	2022-03-01

- `SELECT * FROM employees WHERE salary BETWEEN 50000 AND 70000;`

The screenshot shows the same SQL IDE interface. The query editor now contains:

```
1 • USE demo_db ;
2 • SELECT * FROM employees
3 • WHERE salary BETWEEN 50000 AND 70000;
```

The 'Result Grid' shows the results:

emp_id	name	department	salary	join_date
103	Alice Johnson	Finance	60000	2019-08-23
105	Eve Davis	Marketing	55000	2021-11-05

- `SELECT * FROM employees WHERE name LIKE 'J%';` -- Names starting with J

The screenshot shows the same SQL IDE interface. The query editor now contains:

```
1 • USE demo_db ;
2 • SELECT * FROM employees
3 • WHERE name LIKE 'J%';
```

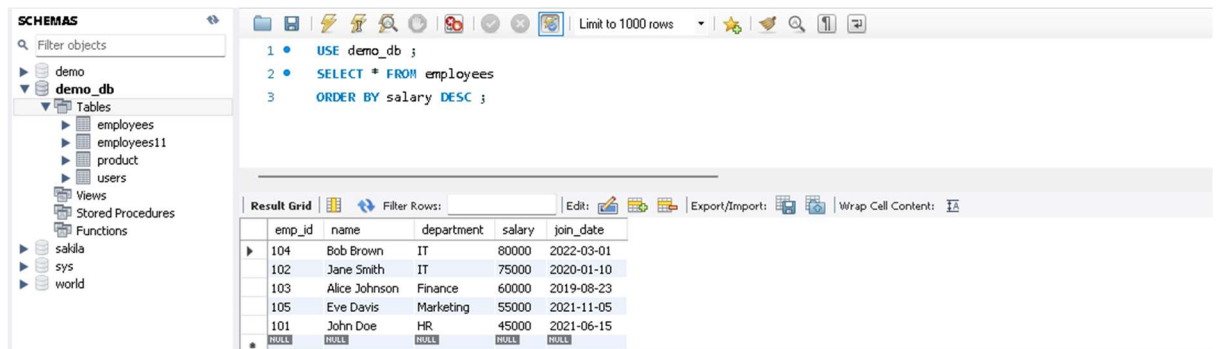
The 'Result Grid' shows the results:

emp_id	name	department	salary	join_date
101	John Doe	HR	45000	2021-06-15
102	Jane Smith	IT	75000	2020-01-10

5) CLAUSE -ORDER BY, WHERE, HAVING

`SELECT * FROM employees`

ORDER BY salary DESC;



The screenshot shows a SQL IDE interface. On the left, the 'SCHEMAS' pane displays a tree view with 'demo_db' expanded, showing tables like 'employees', 'employees11', 'product', and 'users'. The main query editor contains the following SQL code:

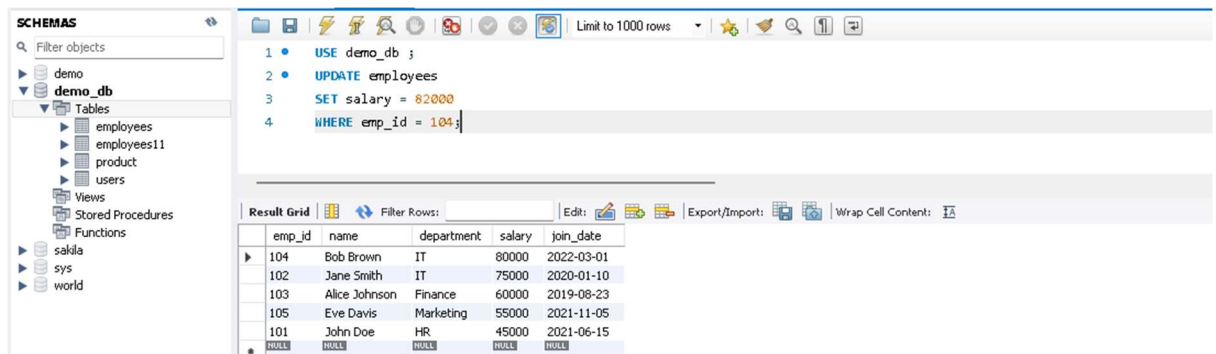
```
1 USE demo_db ;
2 SELECT * FROM employees
3 ORDER BY salary DESC ;
```

Below the query editor, the 'Result Grid' displays the results of the query. The grid has columns: emp_id, name, department, salary, and join_date. The results are ordered by salary in descending order.

emp_id	name	department	salary	join_date
104	Bob Brown	IT	80000	2022-03-01
102	Jane Smith	IT	75000	2020-01-10
103	Alice Johnson	Finance	60000	2019-08-23
105	Eve Davis	Marketing	55000	2021-11-05
101	John Doe	HR	45000	2021-06-15
NULL	NULL	NULL	NULL	NULL

6) UPDATE QUERY

- UPDATE employees
SET salary = 82000
WHERE emp_id = 104;



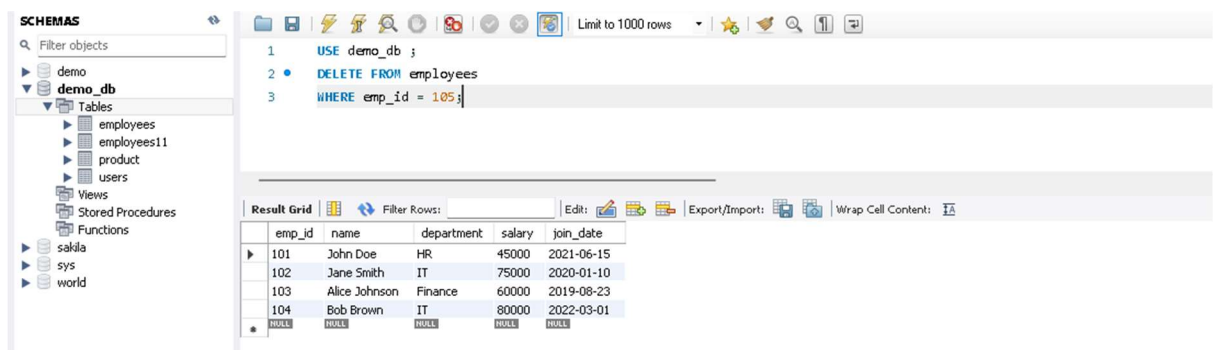
The screenshot shows a SQL IDE interface. The query editor contains the following SQL code:

```
1 USE demo_db ;
2 UPDATE employees
3 SET salary = 82000
4 WHERE emp_id = 104;
```

The 'Result Grid' displays the results of the query. The grid has columns: emp_id, name, department, salary, and join_date. The results are ordered by emp_id in ascending order.

emp_id	name	department	salary	join_date
104	Bob Brown	IT	80000	2022-03-01
102	Jane Smith	IT	75000	2020-01-10
103	Alice Johnson	Finance	60000	2019-08-23
105	Eve Davis	Marketing	55000	2021-11-05
101	John Doe	HR	45000	2021-06-15
NULL	NULL	NULL	NULL	NULL

- DELETE FROM employees
WHERE emp_id = 105;



The screenshot shows a SQL IDE interface. The query editor contains the following SQL code:

```
1 USE demo_db ;
2 DELETE FROM employees
3 WHERE emp_id = 105;
```

The 'Result Grid' displays the results of the query. The grid has columns: emp_id, name, department, salary, and join_date. The results are ordered by emp_id in ascending order.

emp_id	name	department	salary	join_date
101	John Doe	HR	45000	2021-06-15
102	Jane Smith	IT	75000	2020-01-10
103	Alice Johnson	Finance	60000	2019-08-23
104	Bob Brown	IT	80000	2022-03-01
NULL	NULL	NULL	NULL	NULL

- SELECT department, AVG(salary) AS avg_salary

FROM employees
GROUP BY department;

The screenshot shows a SQL IDE interface. On the left, the 'SCHEMAS' pane displays a tree view with 'demo_db' expanded, showing tables like 'employees', 'employees11', 'product', and 'users'. The main editor contains the following SQL query:

```
1 USE demo_db ;
2 SELECT department, AVG(salary)
3 AS avg_salary
4 FROM employees
5 GROUP BY department ;
```

Below the query editor, the 'Result Grid' shows the output of the query:

department	avg_salary
HR	45000
IT	77500
Finance	60000

- SELECT department, COUNT(*) AS emp_count
FROM employees
GROUP BY department
HAVING COUNT(*) > 1;

The screenshot shows the same SQL IDE interface. The 'SCHEMAS' pane is identical. The main editor contains the following SQL query:

```
1 SELECT department, COUNT(*) AS emp_count
2 FROM employees
3 GROUP BY department
4 HAVING COUNT(*) > 1;
```

Below the query editor, the 'Result Grid' shows the output of the query:

department	emp_count
IT	2