

23K-6005

HARIS AHMED

BSAI-5A

DATABASE LAB 04:

IN LAB TASKS

1)

The screenshot shows the Oracle SQL Developer interface. The main window displays a SQL script with four parts. The first part is a COMMIT statement. The second part is a SELECT statement that counts the number of students per department. The third part is a SELECT statement that calculates the average GPA per department, filtering for departments where the average GPA is greater than 3.0. The fourth part is a SELECT statement that counts the number of faculty per department. The script is executed, and the results are shown in the 'Query Result' window at the bottom. The results show three rows: EE with 1 student, CS with 3 students, and Math with 1 student.

```
COMMIT;

--part 1
SELECT d.dept_name, COUNT(s.student_id) AS num_students
FROM Department d
LEFT JOIN Student s ON d.dept_id = s.dept_id
GROUP BY d.dept_name;

--part 2
SELECT d.dept_name, AVG(s.gpa) AS avg_gpa
FROM Department d
JOIN Student s ON d.dept_id = s.dept_id
GROUP BY d.dept_name
HAVING AVG(s.gpa) > 3.0;

--part 3
SELECT c.name AS course_name, AVG(p.fee_amount) AS avg_fee
FROM Course c
JOIN Payment p ON c.course_id = p.course_id
GROUP BY c.name;

--part 4
SELECT d.dept_name, COUNT(f.faculty_id) AS num_faculty
FROM Department d
```

DEPT_NAME	NUM_STUDENTS
1 EE	1
2 CS	3
3 Math	1

2)

Oracle SQL Developer: my_hr_conn

File Edit View Navigate Run Source Team Tools Window Help

Worksheet Query Builder

```

GROUP BY d.dept_name;

--part 2
SELECT d.dept_name, AVG(s.gpa) AS avg_gpa
FROM Department d
JOIN Student s ON d.dept_id = s.dept_id
GROUP BY d.dept_name
HAVING AVG(s.gpa) > 3.0;

--part 3
SELECT c.name AS course_name, AVG(p.fee_amount) AS avg_fee
FROM Course c
JOIN Payment p ON c.course_id = p.course_id
GROUP BY c.name;

--part 4
SELECT d.dept_name, COUNT(f.faculty_id) AS num_faculty
FROM Department d
LEFT JOIN Faculty f ON d.dept_id = f.dept_id
GROUP BY d.dept_name;

--part 5
SELECT f.name, f.salary
FROM Faculty f
WHERE f.salary > (SELECT AVG(salary) FROM Faculty);

```

Script Output x Query Result x

All Rows Fetched: 3 in 0 seconds

DEPT_NAME	NUM_FACULTY
EE	1
CS	3
Math	1

Saved: my_hr_conn

5)

Oracle SQL Developer: my_hr_conn

File Edit View Navigate Run Source Team Tools Window Help

Worksheet Query Builder

```

LEFT JOIN Faculty f ON d.dept_id = f.dept_id
GROUP BY d.dept_name;

--part 5
SELECT f.name, f.salary
FROM Faculty f
WHERE f.salary > (SELECT AVG(salary) FROM Faculty);

--part 6
SELECT s.name, s.gpa
FROM Student s
WHERE s.gpa > ANY (SELECT gpa FROM Student WHERE dept_id = (SELECT dept_id FROM Department WHERE dept_name = 'CS'));

--part 7
SELECT student_id, name, gpa
FROM (SELECT student_id, name, gpa FROM Student ORDER BY gpa DESC)
WHERE ROWNUM <= 3;

--part 8
SELECT s.name
FROM Student s
WHERE NOT EXISTS (
  SELECT course_id FROM Enrollment WHERE student_id = (SELECT student_id FROM Student WHERE name = 'Ali')
  MINUS
);

```

Script Output x Query Result x

All Rows Fetched: 2 in 0 seconds

NAME	SALARY
Prof A	120000
Prof C	150000

Saved: my_hr_conn

6)

Oracle SQL Developer: my_hr_conn

File Edit View Navigate Run Source Team Tools Window Help

Welcome Page my_hr_conn Lab-Tasks.sql

Worksheet Query Builder

```

LEFT JOIN Faculty f ON d.dept_id = f.dept_id
GROUP BY d.dept_name;

--part 5
SELECT f.name, f.salary
FROM Faculty f
WHERE f.salary > (SELECT AVG(salary) FROM Faculty);

--part 6
SELECT s.name, s.gpa
FROM Student s
WHERE s.gpa > ANY (SELECT gpa FROM Student WHERE dept_id = (SELECT dept_id FROM Department WHERE dept_name = 'CS'));

--part 7
SELECT student_id, name, gpa
FROM (SELECT student_id, name, gpa FROM Student ORDER BY gpa DESC)
WHERE ROWNUM <= 3;

--part 8
SELECT s.name
FROM Student s
WHERE NOT EXISTS (
    SELECT course_id FROM Enrollment WHERE student_id = (SELECT student_id FROM Student WHERE name = 'Ali')
    MINUS

```

Script Output x Query Result x

SQL All Rows Fetched: 4 in 0.003 seconds

	NAME	GPA
1	Ali	3.8
2	Charlie	3.2
3	Dana	3.6
4	Eve	4

Saved: my_hr_conn

7)

Oracle SQL Developer: my_hr_conn

File Edit View Navigate Run Source Team Tools Window Help

Welcome Page my_hr_conn Lab-Tasks.sql

Worksheet Query Builder

```

LEFT JOIN Faculty f ON d.dept_id = f.dept_id
GROUP BY d.dept_name;

--part 5
SELECT f.name, f.salary
FROM Faculty f
WHERE f.salary > (SELECT AVG(salary) FROM Faculty);

--part 6
SELECT s.name, s.gpa
FROM Student s
WHERE s.gpa > ANY (SELECT gpa FROM Student WHERE dept_id = (SELECT dept_id FROM Department WHERE dept_name = 'CS'));

--part 7
SELECT student_id, name, gpa
FROM (SELECT student_id, name, gpa FROM Student ORDER BY gpa DESC)
WHERE ROWNUM <= 3;

--part 8
SELECT s.name
FROM Student s
WHERE NOT EXISTS (
    SELECT course_id FROM Enrollment WHERE student_id = (SELECT student_id FROM Student WHERE name = 'Ali')
    MINUS

```

Script Output x Query Result x

SQL All Rows Fetched: 3 in 0.003 seconds

	STUDENT_ID	NAME	GPA
1	5	Eve	4
2	1	Ali	3.8
3	4	Dana	3.6

Saved: my_hr_conn

8)

Oracle SQL Developer: my_hr_conn

File Edit View Navigate Run Source Team Tools Window Help

Worksheet Query Builder

```

FROM (SELECT student_id, name, gpa FROM Student ORDER BY gpa DESC)
WHERE ROWNUM <= 3;

--part 9
SELECT s.name
FROM Student s
WHERE NOT EXISTS (
  SELECT course_id FROM Enrollment WHERE student_id = (SELECT student_id FROM Student WHERE name = 'Ali')
  MINUS
  SELECT course_id FROM Enrollment WHERE student_id = s.student_id
)
AND s.name != 'Ali';

--part 9
SELECT d.dept_name, SUM(p.fee_amount) AS total_fees
FROM Department d
JOIN Payment p ON d.dept_id = p.dept_id
GROUP BY d.dept_name;

--part 10
SELECT DISTINCT c.name AS course_name
FROM Course c
JOIN Enrollment e ON c.course_id = e.course_id
JOIN Student s ON e.student_id = s.student_id
WHERE s.gpa > 3.5;

```

Script Output x Query Result x

All Rows Fetched: 1 in 0 seconds

NAME
1 Dana

Saved: my_hr_conn

9)

Oracle SQL Developer: my_hr_conn

File Edit View Navigate Run Source Team Tools Window Help

Worksheet Query Builder

```

AND s.name != 'Ali';

--part 9
SELECT d.dept_name, SUM(p.fee_amount) AS total_fees
FROM Department d
JOIN Payment p ON d.dept_id = p.dept_id
GROUP BY d.dept_name;

--part 10
SELECT DISTINCT c.name AS course_name
FROM Course c
JOIN Enrollment e ON c.course_id = e.course_id
JOIN Student s ON e.student_id = s.student_id
WHERE s.gpa > 3.5;

--part 11
SELECT d.dept_name, SUM(p.fee_amount) AS total_fees
FROM Department d
JOIN Payment p ON d.dept_id = p.dept_id
GROUP BY d.dept_name
HAVING SUM(p.fee_amount) > 1000000;

--part 12
SELECT d.dept_name

```

Script Output x Query Result x

All Rows Fetched: 3 in 0.004 seconds

DEPT_NAME	TOTAL_FEES
1 EE	3000
2 CS	26000
3 Math	4000

Saved: my_hr_conn

10)

```

AND s.name != 'AI1';
--part 9
SELECT d.dept_name, SUM(p.fee_amount) AS total_fees
FROM Department d
JOIN Payment p ON d.dept_id = p.dept_id
GROUP BY d.dept_name;
--part 10
SELECT DISTINCT c.name AS course_name
FROM Course c
JOIN Enrollment e ON c.course_id = e.course_id
JOIN Student s ON e.student_id = s.student_id
WHERE s.gpa > 3.5;
--part 11
SELECT d.dept_name, SUM(p.fee_amount) AS total_fees
FROM Department d
JOIN Payment p ON d.dept_id = p.dept_id
GROUP BY d.dept_name
HAVING SUM(p.fee_amount) > 1000000;
--part 12
SELECT d.dept_name

```

Script Output x Query Result x

SQL All Rows Fetched: 3 in 0.004 seconds

COURSE_NAME
1 Algebra
2 Databases
3 AI

Saved: my_hr_conn | Line 197 Column 15 | Insert | Windows: C

POST LAB TASKS

11)

```

AND s.name != 'AI1';
--part 9
SELECT d.dept_name, SUM(p.fee_amount) AS total_fees
FROM Department d
JOIN Payment p ON d.dept_id = p.dept_id
GROUP BY d.dept_name;
--part 10
SELECT DISTINCT c.name AS course_name
FROM Course c
JOIN Enrollment e ON c.course_id = e.course_id
JOIN Student s ON e.student_id = s.student_id
WHERE s.gpa > 3.5;
--part 11
SELECT d.dept_name, SUM(p.fee_amount) AS total_fees
FROM Department d
JOIN Payment p ON d.dept_id = p.dept_id
GROUP BY d.dept_name
HAVING SUM(p.fee_amount) > 1000000;
--part 12
SELECT d.dept_name

```

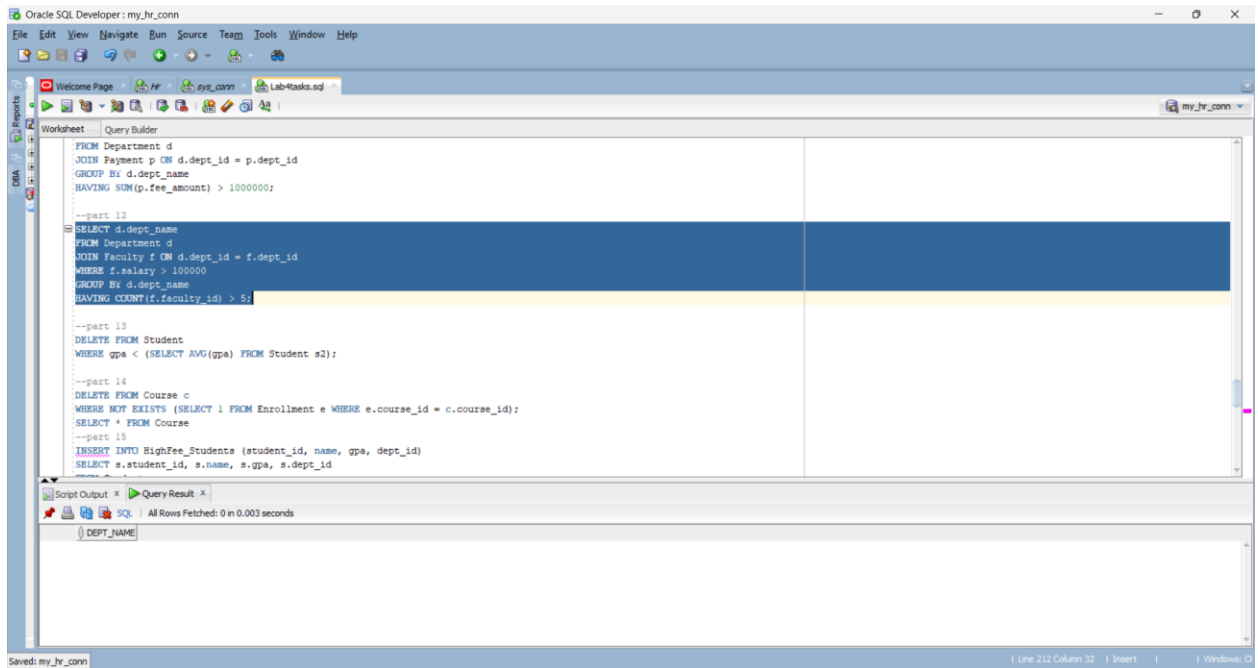
Script Output x Query Result x

SQL All Rows Fetched: 0 in 0.002 seconds

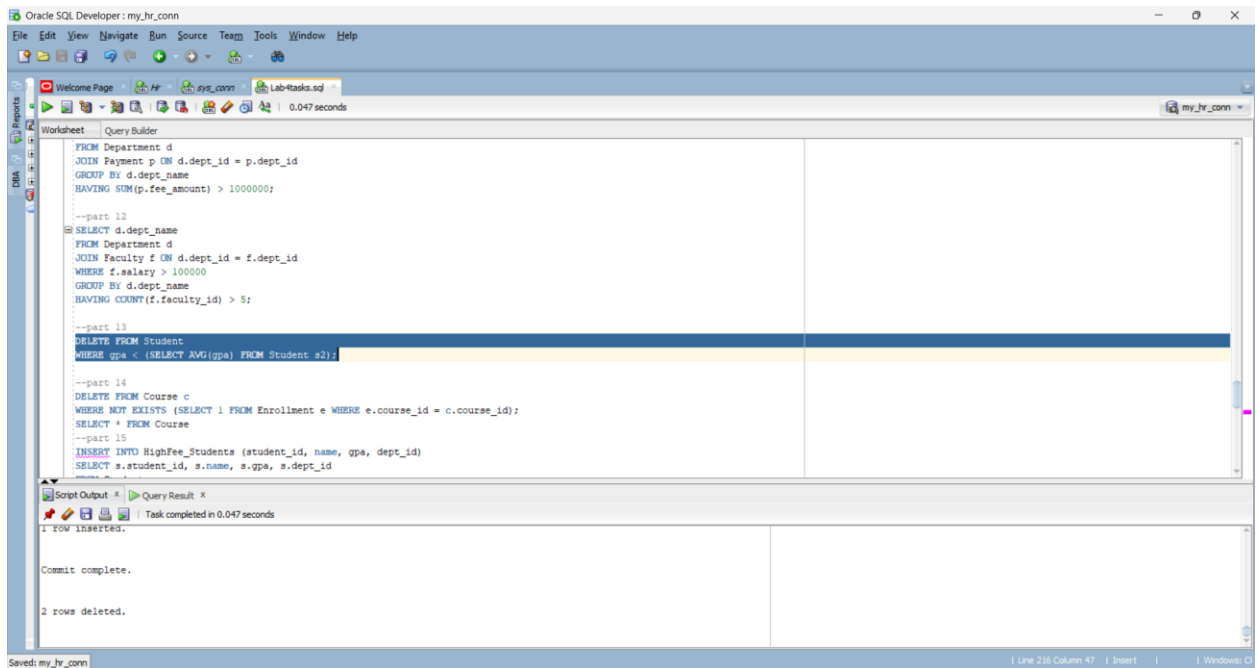
DEPT_NAME	TOTAL_F...
-----------	------------

Saved: my_hr_conn | Line 204 Column 36 | Insert | Windows: C

12)



13)



14)

Oracle SQL Developer: my_hr_conn

File Edit View Navigate Run Source Team Tools Window Help

Worksheet

```

FROM Department d
JOIN Payment p ON d.dept_id = p.dept_id
GROUP BY d.dept_name
HAVING SUM(p.fee_amount) > 1000000;

--part 12
SELECT d.dept_name
FROM Department d
JOIN Faculty f ON d.dept_id = f.dept_id
WHERE f.salary > 100000
GROUP BY d.dept_name
HAVING COUNT(f.faculty_id) > 5;

--part 13
DELETE FROM Student
WHERE gpa < (SELECT AVG(gpa) FROM Student s2);

--part 14
DELETE FROM Course c
WHERE NOT EXISTS (SELECT 1 FROM Enrollment e WHERE e.course_id = c.course_id);
SELECT * FROM Course;

--part 15
INSERT INTO HighFee_Students (student_id, name, gpa, dept_id)
SELECT s.student_id, s.name, s.gpa, s.dept_id

```

Script Output | Query Result | Query Result 1

All Rows Fetched: 3 in 0 seconds

COURSE_ID	NAME	DEPT_ID	FACULTY_ID
1	Databases	1	1
2	Algebra	3	4
3	AI	1	3

Saved: my_hr_conn

15)

Oracle SQL Developer: my_hr_conn

File Edit View Navigate Run Source Team Tools Window Help

Worksheet

```

SELECT * FROM Course;

--part 15
INSERT INTO HighFee_Students (student_id, name, gpa, dept_id)
SELECT s.student_id, s.name, s.gpa, s.dept_id
FROM Student s
JOIN Payment p ON s.student_id = p.student_id
GROUP BY s.student_id, s.name, s.gpa, s.dept_id
HAVING SUM(p.fee_amount) > (SELECT AVG(fee_amount) FROM Payment);
SELECT * FROM HighFee_Students;

--part 16
INSERT INTO Retired_Faculty (faculty_id, name, salary, joining_date, dept_id)
SELECT faculty_id, name, salary, joining_date, dept_id
FROM Faculty
WHERE joining_date < (SELECT MIN(joining_date) FROM Faculty);

--part 17
SELECT dept_name, total_fees
FROM (
    SELECT d.dept_name, SUM(p.fee_amount) AS total_fees,
    RANK() OVER (ORDER BY SUM(p.fee_amount) DESC) AS rnk
    FROM Department d
    JOIN Payment p ON d.dept_id = p.dept_id
    GROUP BY d.dept_name
)

```

Script Output | Query Result | Query Result 1 | Query Result 2

All Rows Fetched: 2 in 0.006 seconds

STUDENT_ID	NAME	GPA	DEPT_ID
1	Ali	3.8	1
2	Dana	3.6	1

Saved: my_hr_conn

16)

Oracle SQL Developer: my_hr_conn

File Edit View Navigate Run Source Team Tools Window Help

Worksheet Query Builder

```

--part 15
SELECT * FROM course
INSERT INTO HighFee_Students (student_id, name, gpa, dept_id)
SELECT s.student_id, s.name, s.gpa, s.dept_id
FROM Student s
JOIN Payment p ON s.student_id = p.student_id
GROUP BY s.student_id, s.name, s.gpa, s.dept_id
HAVING SUM(p.fee_amount) > (SELECT AVG(fee_amount) FROM Payment);
SELECT * FROM HighFee_Students

--part 16
INSERT INTO Retired_Faculty (faculty_id, name, salary, joining_date, dept_id)
SELECT faculty_id, name, salary, joining_date, dept_id
FROM Faculty
WHERE joining_date < (SELECT MIN(joining_date) FROM Faculty);

--part 17
SELECT dept_name, total_fees
FROM (
    SELECT d.dept_name, SUM(p.fee_amount) AS total_fees,
    RANK() OVER (ORDER BY SUM(p.fee_amount) DESC) AS rnk
    FROM Department d
    JOIN Payment p ON d.dept_id = p.dept_id
    GROUP BY d.dept_name
)

```

Script Output X Query Result X Query Result 1 X Query Result 2 X

Task completed in 0.032 seconds

>>Query Run In:Query Result 1

2 rows inserted.

>>Query Run In:Query Result 2

0 rows inserted.

Saved: my_hr_conn

17)

Oracle SQL Developer: my_hr_conn

File Edit View Navigate Run Source Team Tools Window Help

Worksheet Query Builder

```

--part 16
INSERT INTO Retired_Faculty (faculty_id, name, salary, joining_date, dept_id)
SELECT faculty_id, name, salary, joining_date, dept_id
FROM Faculty
WHERE joining_date < (SELECT MIN(joining_date) FROM Faculty);

--part 17
SELECT dept_name, total_fees
FROM (
    SELECT d.dept_name, SUM(p.fee_amount) AS total_fees,
    RANK() OVER (ORDER BY SUM(p.fee_amount) DESC) AS rnk
    FROM Department d
    JOIN Payment p ON d.dept_id = p.dept_id
    GROUP BY d.dept_name
)
WHERE rnk = 1;

--part 18
SELECT course_name, num_enrollments
FROM (
    SELECT c.name AS course_name, COUNT(e.enrollment_id) AS num_enrollments
    FROM Course c
    LEFT JOIN Enrollment e ON c.course_id = e.course_id
)

```

Script Output X Query Result X Query Result 1 X Query Result 2 X Query Result 3 X

All Rows Fetched: 1 in 0 seconds

DEPT_NAME	TOTAL_FEE
1 CS	22000

Saved: my_hr_conn

18)

Oracle SQL Developer: my_hr_conn

File Edit View Navigate Run Source Team Tools Window Help

Worksheet Query Builder

```

GROUP BY d.dept_name
)
WHERE rnk = 1;

--part 18
SELECT course_name, num_enrollments
FROM (
  SELECT c.name AS course_name, COUNT(e.enrollment_id) AS num_enrollments
  FROM Course c
  LEFT JOIN Enrollment e ON c.course_id = e.course_id
  GROUP BY c.name
  ORDER BY num_enrollments DESC
)
WHERE ROWNUM <= 3;

--part 19
SELECT s.name, s.gpa
FROM Student s
WHERE s.gpa > (SELECT AVG(gpa) FROM Student)
AND s.student_id IN (
  SELECT student_id
  FROM Enrollment
  GROUP BY student_id
  HAVING COUNT(course_id) > 3
)

```

Script Output x Query Result 1 x Query Result 2 x Query Result 3 x

All Rows Fetched: 3 in 0 seconds

COURSE_NAME	NUM_ENROLLMENTS
1 Algebra	3
2 Databases	2
3 AI	2

Saved: my_hr_conn

19)

Oracle SQL Developer: my_hr_conn

File Edit View Navigate Run Source Team Tools Window Help

Worksheet Query Builder

```

FROM Course c
LEFT JOIN Enrollment e ON c.course_id = e.course_id
GROUP BY c.name
ORDER BY num_enrollments DESC
)
WHERE ROWNUM <= 3;

--part 19
SELECT s.name, s.gpa
FROM Student s
WHERE s.gpa > (SELECT AVG(gpa) FROM Student)
AND s.student_id IN (
  SELECT student_id
  FROM Enrollment
  GROUP BY student_id
  HAVING COUNT(course_id) > 3
)
);

--part 20
INSERT INTO Unassigned_Faculty (faculty_id, name, salary, dept_id)
SELECT f.faculty_id, f.name, f.salary, f.dept_id
FROM Faculty f
WHERE NOT EXISTS (SELECT 1 FROM Course c WHERE c.faculty_id = f.faculty_id);
SELECT * FROM Unassigned_Faculty

```

Script Output x Query Result 1 x Query Result 2 x Query Result 3 x

All Rows Fetched: 0 in 0.005 seconds

NAME	GPA
------	-----

Saved: my_hr_conn

20)

The screenshot shows the Oracle SQL Developer interface with a script named 'my_hr_conn'. The script contains several SQL queries. The first query is a SELECT statement that joins the 'Course' and 'Enrollment' tables, groups by course name, and orders by the number of enrollments in descending order. The second query is a SELECT statement that filters students based on their GPA and enrollment in specific courses. The third query is an INSERT statement that adds new faculty members to the 'Unassigned_Faculty' table based on a specific condition.

```
FROM Course c
LEFT JOIN Enrollment e ON c.course_id = e.course_id
GROUP BY c.name
ORDER BY num_enrollments DESC
)
WHERE ROWNUM <= 3;

--part 19
SELECT s.name, s.gpa
FROM Student s
WHERE s.gpa > (SELECT AVG(gpa) FROM Student)
AND s.student_id IN (
SELECT student_id
FROM Enrollment
GROUP BY student_id
HAVING COUNT(course_id) > 3
);

--part 20
INSERT INTO Unassigned_Faculty (faculty_id, name, salary, dept_id)
SELECT f.faculty_id, f.name, f.salary, f.dept_id
FROM Faculty f
WHERE NOT EXISTS (SELECT 1 FROM Course c WHERE c.faculty_id = f.faculty_id);
SELECT * FROM Unassigned_Faculty;
```

The execution results are displayed in a table with the following columns: FACULTY_ID, NAME, SALARY, and DEPT_ID. The table contains two rows of data.

FACULTY_ID	NAME	SALARY	DEPT_ID
1	5 Prof E	110000	1
2	2 Prof B	90000	2