

Name: Mat Aisas

Class: ITE M2

WCT II

Part 5: Insert Sample Data

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'SCHEMAS' pane displays a tree view of the database objects, including 'Tables', 'Views', 'Stored Procedures', and 'Functions'. The 'student' table is selected under 'Tables'. The 'Table: student' information pane shows the following columns:

Column	DataType	PK
student_id	int	PK
first_name	varchar(45)	
last_name	varchar(45)	
email	varchar(45)	
date_of_birth	date	

The main query window shows the following SQL script:

```
1 USE wct_ass3;  
2 INSERT INTO student (student_id,first_name, last_name, date_of_birth, email)  
3 VALUES ('1','Alice', 'Johnson', '2001-06-15', 'alice@example.com');
```

The 'Output' pane at the bottom shows the execution results:

#	Time	Action	Message
1	10:05:14	INSERT INTO student (student_id,first_name, last_name, date_of_birth, email) VALUE...	1 row(s) affected

Part 6: Querying the Database Write SQL queries to answer:

1. Retrieve all students who enrolled in a specific course.

The screenshot shows the SQL Server Enterprise Manager interface. The 'student' table is selected in the 'SCHEMAS' pane. The main query window shows the following SQL script:

```
1 USE wct_ass3;  
2 INSERT INTO student (student_id, first_name, last_name, date_of_birth, email)  
3 VALUES ('1','Alice', 'Johnson', '2001-06-15', 'alice@example.com');  
4  
5 SELECT s.student_id, s.first_name, s.last_name, s.email  
6 FROM student s  
7 JOIN enrollment e ON s.student_id = e.student_id  
8 JOIN course c ON e.course_id = c.course_id  
9 WHERE c.course_code = 'CS101';
```

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

student_id	first_name	last_name	email
1	Alice	Johnson	alice@example.com

Explanation:

- The students table contains student details.
- The enrollments table links students and courses.
- The courses table contains course details.
- We use JOIN to connect the three tables using student_id (from students → enrollments) and course_id (from enrollments → courses).
- WHERE c.course_code = 'CS101' filters the results to show only students enrolled in CS101.

2. Find all faculty members in a particular department

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Filter objects' pane shows a tree view of database objects, including 'Tables' (course, department, enrollment, faculty, student), 'Views', 'Stored Procedures', and 'Functions'. The 'Information' pane shows the structure of the 'faculty' table: faculty_id (int PK), name (varchar), email (varchar), title (varchar), and department_id (int). The main query editor displays the following SQL code:

```
6 FROM student s
7 JOIN enrollment e ON s.student_id = e.student_id
8 JOIN course c ON e.course_id = c.course_id
9 WHERE c.course_code = 'CS101';
10
11 • SELECT f.faculty_id, f.name, f.email, f.title
12 FROM faculty f
13 JOIN department d ON f.department_id = d.department_id
14 WHERE d.department_name = 'Computer Science';
```

The 'Result Grid' pane shows the columns: faculty_id, name, email, title.

Explanation:

- The faculty table contains faculty details.
- The departments table contains department details.
- We join these tables using department_id since each faculty member belongs to one department.
- WHERE d.department_name = 'Computer Science' filters the results to show only faculty members in the Computer Science department.

3. List all courses a particular student is enrolled in.

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Filter objects' pane shows a tree view of database objects, including 'Tables' (course, department, enrollment, faculty, student), 'Views', 'Stored Procedures', and 'Functions'. The 'Information' pane shows the structure of the 'course' table: course_id (int PK), course_code (int), course_name (varchar), credits (varchar), faculty_id (int), and department_id (int). The main query editor displays the following SQL code:

```
15
16
17
18 • SELECT c.course_code, c.course_name, c.credits
19 FROM course c
20 JOIN enrollment e ON c.course_id = e.course_id
21 JOIN student s ON e.student_id = s.student_id
22 WHERE s.first_name = 'Alice' AND s.last_name = 'Johnson';
23
```

The 'Result Grid' pane shows the columns: course_code, course_name, credits.

Explanation:

- The students table provides student names.
- The enrollments table links students to courses.
- The courses table provides course details.
- We use JOIN to connect these tables.
- WHERE s.first_name = 'Alice' AND s.last_name = 'Johnson' ensures we retrieve courses for Alice Johnson only.

4. Retrieve students who have not enrolled in any course.

The screenshot shows a database management interface. On the left, a 'SCHEMAS' panel lists tables: course, department, enrollment, faculty, student, and Views. The 'course' table is selected, showing its columns: course_id (int PK), course_code (int), course_name (varchar), credits (varchar), faculty_id (int), and department_id (int). The main area displays a SQL query:

```
22 WHERE s.first_name = 'Alice' AND s.last_name = 'Johnson';
23
24
25
26 • SELECT s.student_id, s.first_name, s.last_name, s.email
27 FROM student s
28 LEFT JOIN enrollment e ON s.student_id = e.student_id
29 WHERE e.enrollment_id IS NULL;
30
```

Below the query, the 'Result Grid' shows one row of data:

student_id	first_name	last_name	email
1	Alice	Johnson	alice@example.com

The interface also includes a 'Filter Rows' section, an 'Export' button, and a 'Wrap Cell Content' option. The bottom of the screen shows an 'Output' section with an 'Action Output' dropdown and a 'Message' field.

Explanation:

- We use a LEFT JOIN between students and enrollments to keep all students, even those without enrollments.
- If a student has never enrolled, their student_id will have no matching entry in enrollments, meaning e.enrollment_id will be NULL.
- WHERE e.enrollment_id IS NULL filters out students who have enrolled in at least one course, showing only those who haven't.

5. Find the average grade of students in a specific course.

The screenshot shows a database management tool interface. On the left, the 'SCHEMAS' pane displays a tree view of database objects: Tables (course, department, enrollment, faculty, student), Views, Stored Procedures, and Functions. The 'course' table is selected, and its details are shown in the 'Information' pane below. The 'course' table has columns: course_id (int PK), course_code (int), course_name (varchar), credits (varchar), faculty_id (int), and department_id (int). The main editor displays a SQL query:

```

33 SELECT c.course_code, c.course_name, AVG(
34     CASE
35         WHEN grade = 'A' THEN 4.0
36         WHEN grade = 'B' THEN 3.0
37         WHEN grade = 'C' THEN 2.0
38         WHEN grade = 'D' THEN 1.0
39         WHEN grade = 'F' THEN 0.0
40         ELSE NULL
41     END) AS average_gpa
42 FROM enrollment e
43 JOIN course c ON e.course_id = c.course_id
44 WHERE c.course_code = 'CS101'
45 GROUP BY c.course_id;
46

```

At the bottom, the 'Result Grid' shows the output of the query:

course_code	course_name	average_gpa

Explanation:

- The enrollments table contains student grades.
- The courses table contains course details.
- We use JOIN to connect courses to enrollments.
- The CASE statement converts letter grades into GPA values (A = 4.0, B = 3.0, etc.).
- AVG(...) calculates the average GPA for that course.
- WHERE c.course_code = 'CS101' filters only for CS101.
- GROUP BY c.course_id ensures we calculate one average per course.