ADVANCED DATABASE TOPICS – (COMP-8207)

ASSIGNMENT 2

Name: Ruchi Nakum

Student Id: 110122972

Section: 3

1. Database Schema Design

I. `program` (programID, name)

```
-- Create a table 'program' to store information about academic programs
CREATE TABLE program (
    programID INT PRIMARY KEY,  -- Unique identifier for each program
    name NVARCHAR(255) NOT NULL -- Name of the program
);
```

II. `depCourse` (courseID, deptName, programID)

```
-- Create a table 'depCourse' to store information about departmental courses
CREATE TABLE depCourse (
    courseID INT PRIMARY KEY,  --Unique identifier for each course
    deptName NVARCHAR(255) NOT NULL,  -- Name of the department
    programID INT, -- Foreign key linking to the 'program' table
    FOREIGN KEY (programID) REFERENCES program(programID)  -- Establishing a foreign key relationship
);
```

III. `users` (userID, programID)

```
--Create a table 'users' to store information about users and their associated programs
CREATE TABLE users (
    userID INT PRIMARY KEY,  -- Unique identifier for each user
    programID INT,  -- Foreign key linking to the 'program' table
    FOREIGN KEY (programID) REFERENCES program(programID) -- Establishing a foreign key relationship
);
```

IV. `courseSiteVisit` (visitID, courseID, userID, date)

```
-- Create a table 'courseSiteVisit' to store information about visits to courses
CREATE TABLE courseSiteVisit (
    visitID INT PRIMARY KEY, -- Unique identifier for each visit
    courseID INT, -- Foreign key linking to the 'depCourse' table
    userID INT, -- Foreign key linking to the 'users' table
    date DATE, -- Date of the visit
    FOREIGN KEY (courseID) REFERENCES depCourse(courseID),   -- Establishing a foreign key relationship
    FOREIGN KEY (userID) REFERENCES users(userID) -- Establishing a foreign key relationship
);
```

2. Data Population

i. Inserting data into the 'program' table.

```
--Insert sample data into the 'program' table
INSERT INTO program (programID, name) VALUES
(1, 'PhD'),
(2, 'Master');
```

Output:

	programID	name
1	1	PhD
2	2	Master

ii. Inserting data into the 'depCourse' table.

```
-- Insert sample data into the 'depCourse' table
INSERT INTO depCourse (courseID, deptName, programID) VALUES
(1, 'Networking', 1),
(2, 'Networking', 2),
(3, 'Systems Programming', 1),
(4, 'Systems Programming', 2);

-- Show all courses in the 'depCourse' table after sample data insertion
select * from depCourse;
```

Output:

	courseID	deptName	programID
1	1	Networking	1
2	2	Networking	2
3	3	Systems Programming	1
4	4	Systems Programming	2

iii. Inserting data into the ‘users’ table.

```
--Insert sample data into the 'users' table
INSERT INTO users (userID, programID) VALUES
(1, 1),
(2, 1),
(3, 1),
(4, 2),
(5, 2),
(6, 2);

-- Show all courses in the 'users' table after sample data insertion
select * from users;
```

Output:

	userID	programID
1	1	1
2	2	1
3	3	1
4	4	2
5	5	2
6	6	2

iv. Inserting data into ‘courseSiteVisit’ table.

```
DECLARE @r INT = 1;

WHILE @r <= 100
BEGIN
    INSERT INTO courseSiteVisit (visitID, courseID, userID, date) VALUES
    (@r, 1, (SELECT TOP 1 userID FROM users ORDER BY NEWID()), DATEADD(day, ABS(CHECKSUM(NEWID())) % 30, '2023-05-10')),
    (@r + 1, 2, (SELECT TOP 1 userID FROM users ORDER BY NEWID()), DATEADD(day, ABS(CHECKSUM(NEWID())) % 30, '2023-05-10')),
    (@r + 2, 3, (SELECT TOP 1 userID FROM users ORDER BY NEWID()), DATEADD(day, ABS(CHECKSUM(NEWID())) % 30, '2023-05-10')),
    (@r + 3, 4, (SELECT TOP 1 userID FROM users ORDER BY NEWID()), DATEADD(day, ABS(CHECKSUM(NEWID())) % 30, '2023-05-10')),
    (@r + 4, 1, (SELECT TOP 1 userID FROM users ORDER BY NEWID()), DATEADD(day, ABS(CHECKSUM(NEWID())) % 30, '2023-05-10')),
    (@r + 5, 2, (SELECT TOP 1 userID FROM users ORDER BY NEWID()), DATEADD(day, ABS(CHECKSUM(NEWID())) % 30, '2023-05-10')),
    (@r + 6, 3, (SELECT TOP 1 userID FROM users ORDER BY NEWID()), DATEADD(day, ABS(CHECKSUM(NEWID())) % 30, '2023-05-10'));

    SET @r = @r + 7;
END;
```

Output:

	visitID	courseID	userID	date
1	1	1	6	2023-06-01
2	2	2	1	2023-05-23
3	3	3	6	2023-05-27
4	4	4	2	2023-05-29
5	5	1	6	2023-06-02
6	6	2	3	2023-05-29
7	7	3	4	2023-05-18
8	8	1	4	2023-06-01
9	9	2	2	2023-05-26
10	10	3	6	2023-05-13
11	11	4	5	2023-05-21
12	12	1	1	2023-06-07
13	13	2	6	2023-05-29
14	14	3	4	2023-05-10
15	15	1	6	2023-05-21

3. Data Analysis and Visualization

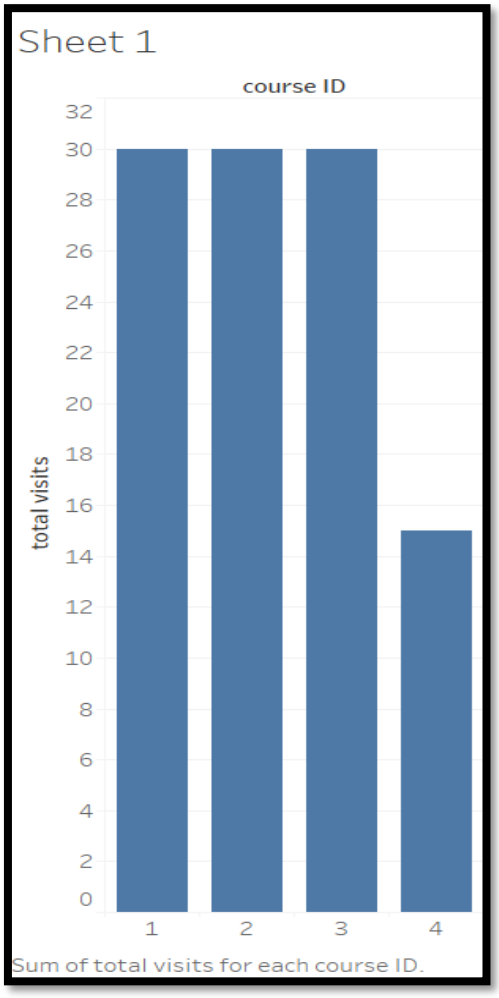
a. The total number of times a course has been visited or accessed by all users

```
--- 1. Count the total visits for each course
SELECT
    courseID,
    COUNT(visitID) AS TotalVisits
FROM
    courseSiteVisit
GROUP BY
    courseID;
```

Output:

	courseID	TotalVisits
1	1	30
2	2	30
3	3	30
4	4	15

Visualization:



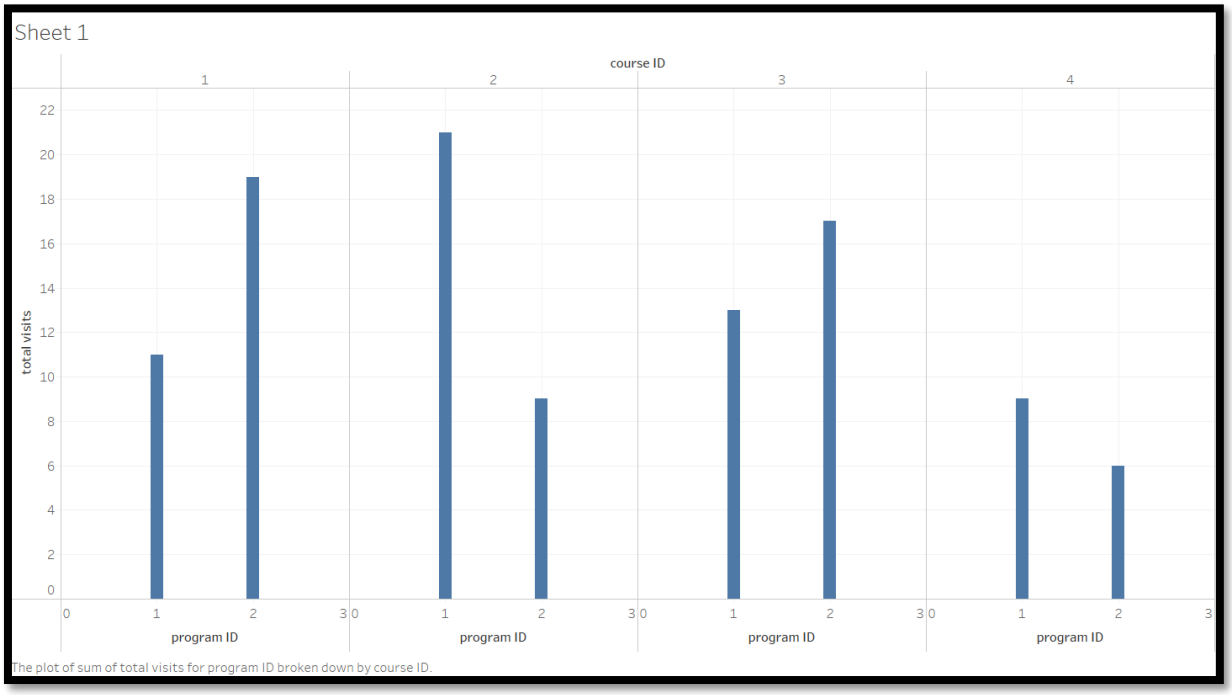
b. The total number of visits for each course, categorized by program

```sql
--- 2. Count the total visits for each course, categorized by program
SELECT
    dc.courseID,
    p.programID,
    COUNT(cv.visitID) AS TotalVisits
FROM
    courseSiteVisit cv
JOIN
    users u ON cv.userID = u.userID
JOIN
    depCourse dc ON cv.courseID = dc.courseID
JOIN
    program p ON u.programID = p.programID
GROUP BY
    dc.courseID, p.programID;
```

Output:

	courseID	programID	TotalVisits
1	1	1	11
2	2	1	21
3	3	1	13
4	4	1	9
5	1	2	19
6	2	2	9
7	3	2	17
8	4	2	6

Visualization:



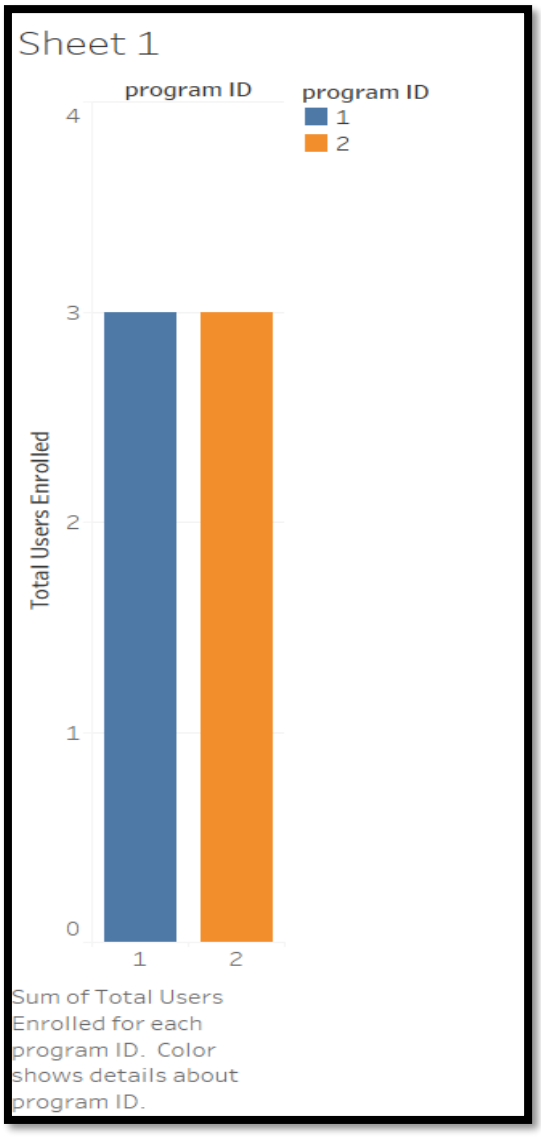
c. The total number of students or users enrolled in each program

```sql
---3. Count the total number of students enrolled in each program
SELECT
    p.programID,
    COUNT(u.userID) AS TotalUsersEnrolled
FROM
    program p
LEFT JOIN
    users u ON p.programID = u.programID
GROUP BY
    p.programID;
```

Output:

	programID	TotalUsersEnrolled
1	1	3
2	2	3

Visualization:



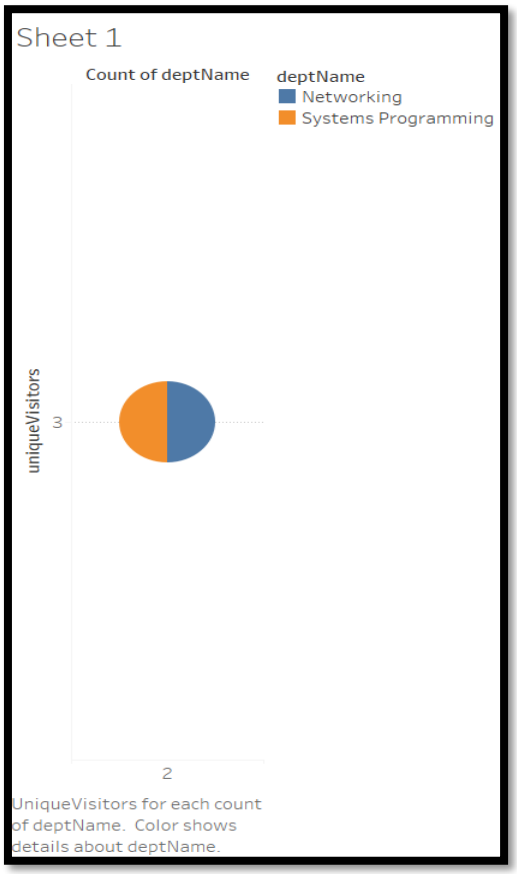
d. The total number of unique visitors per department by program

```sql
---4. Count the total number of unique visitors per department by program
SELECT
    dc.deptName,
    p.programID,
    COUNT(DISTINCT cv.userID) AS UniqueVisitors
FROM
    courseSiteVisit cv
JOIN
    users u ON cv.userID = u.userID
JOIN
    depCourse dc ON cv.courseID = dc.courseID
JOIN
    program p ON u.programID = p.programID
GROUP BY
    dc.deptName, p.programID;
```

Output:

	deptName	programID	UniqueVisitors
1	Networking	1	3
2	Systems Programming	1	3
3	Networking	2	3
4	Systems Programming	2	3

Visualization:



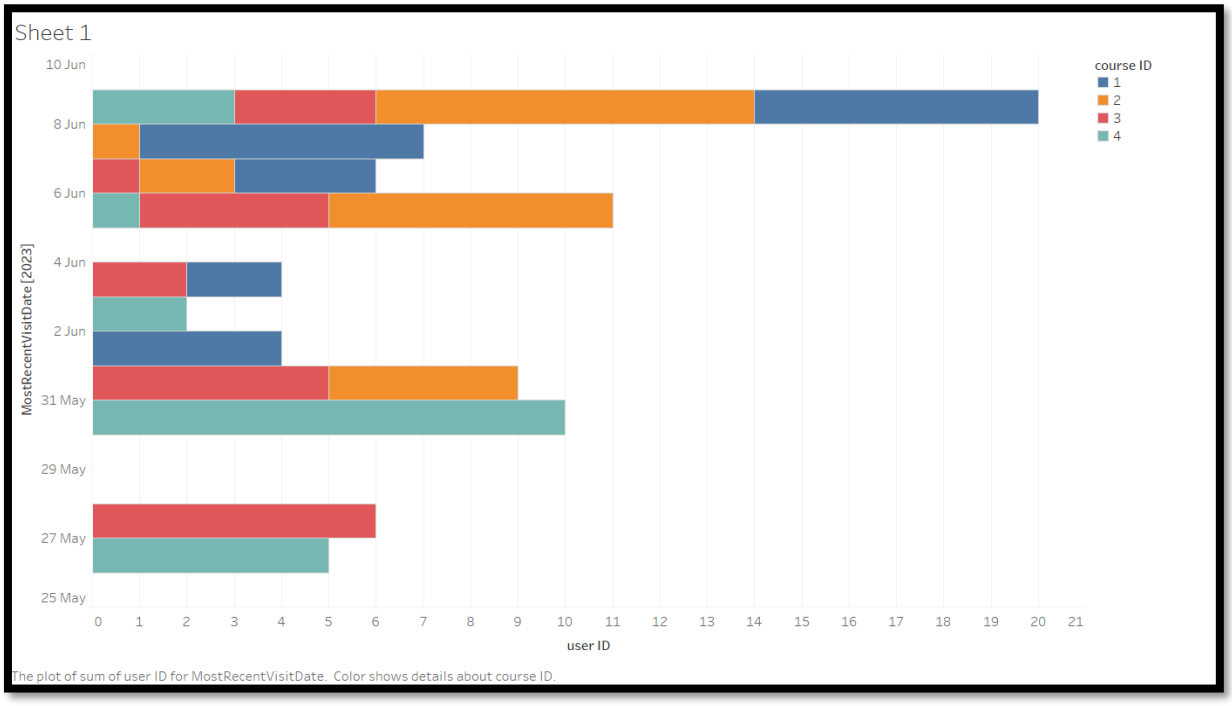
e. The most recent date (or last date) on which a user visited each course

```
---5. Find the most recent visit date for each user and course
SELECT
    cv.courseID,
    cv.userID,
    MAX(cv.date) AS MostRecentVisitDate
FROM
    courseSiteVisit cv
GROUP BY
    cv.courseID, cv.userID;
```

Output:

	courseID	userID	MostRecentVisitDate
1	1	1	2023-06-08
2	2	1	2023-06-07
3	3	1	2023-06-06
4	4	1	2023-06-05
5	1	2	2023-06-03
6	2	2	2023-06-06
7	3	2	2023-06-03
8	4	2	2023-06-02
9	1	3	2023-06-06
10	2	3	2023-06-08
11	3	3	2023-06-08
12	4	3	2023-06-08
13	1	4	2023-06-01
14	2	4	2023-05-31
15	3	4	2023-06-05
16	4	4	2023-05-30
17	1	5	2023-06-08
18	2	5	2023-06-08
19	3	5	2023-05-31
20	4	5	2023-05-26

Visualization:



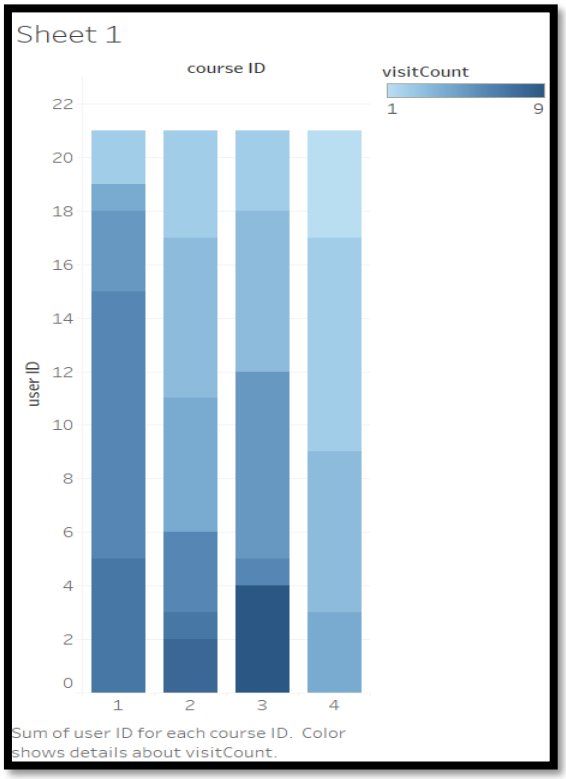
f. The number of times a user has visited each course

```sql
---6. Count the number of times each user visited each course
SELECT
    cv.courseID,
    cv.userID,
    COUNT(cv.visitID) AS VisitCount
FROM
    courseSiteVisit cv
GROUP BY
    cv.courseID, cv.userID;
```

Output:

	courseID	userID	VisitCount
1	1	1	4
2	2	1	7
3	3	1	6
4	4	1	3
5	1	2	2
6	2	2	8
7	3	2	5
8	4	2	2
9	1	3	5
10	2	3	6
11	3	3	2
12	4	3	4
13	1	4	6
14	2	4	2
15	3	4	9
16	4	4	1
17	1	5	7
18	2	5	4
19	3	5	5
20	4	5	3

Visualization:



g. The user who has visited a course the most (i.e., most frequent visitor per course), along with the visit count

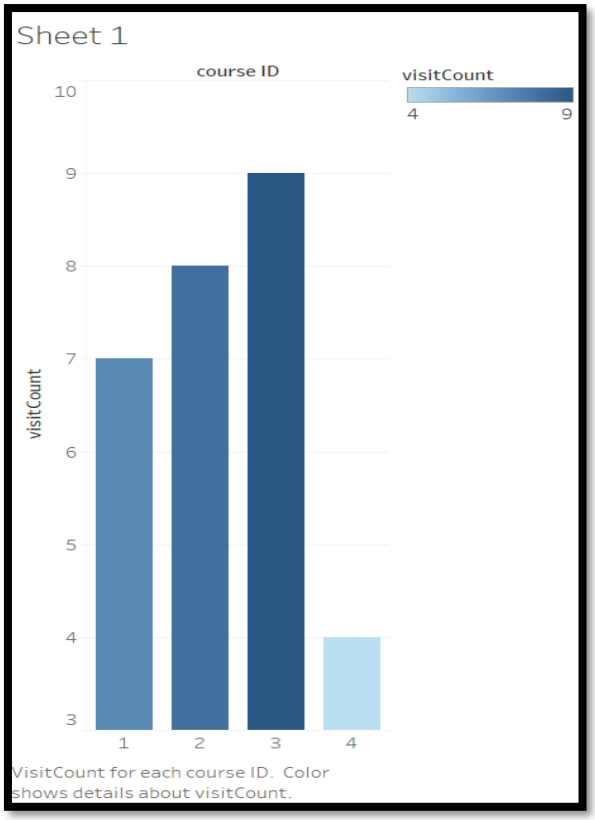
```
---7. Identify the user who visited each course the most
WITH CourseVisitCounts AS (
    SELECT
        cv.courseID,
        cv.userID,
        COUNT(cv.visitID) AS VisitCount,
        ROW_NUMBER() OVER (PARTITION BY cv.courseID ORDER BY COUNT(cv.visitID) DESC) AS Rank
    FROM
        courseSiteVisit cv
    GROUP BY
        cv.courseID, cv.userID
)

SELECT
    courseID,
    userID,
    VisitCount
FROM
    CourseVisitCounts
WHERE
    Rank = 1;
```

Output:

	courseID	userID	VisitCount
1	1	5	7
2	2	2	8
3	3	4	9
4	4	3	4

Visualization:



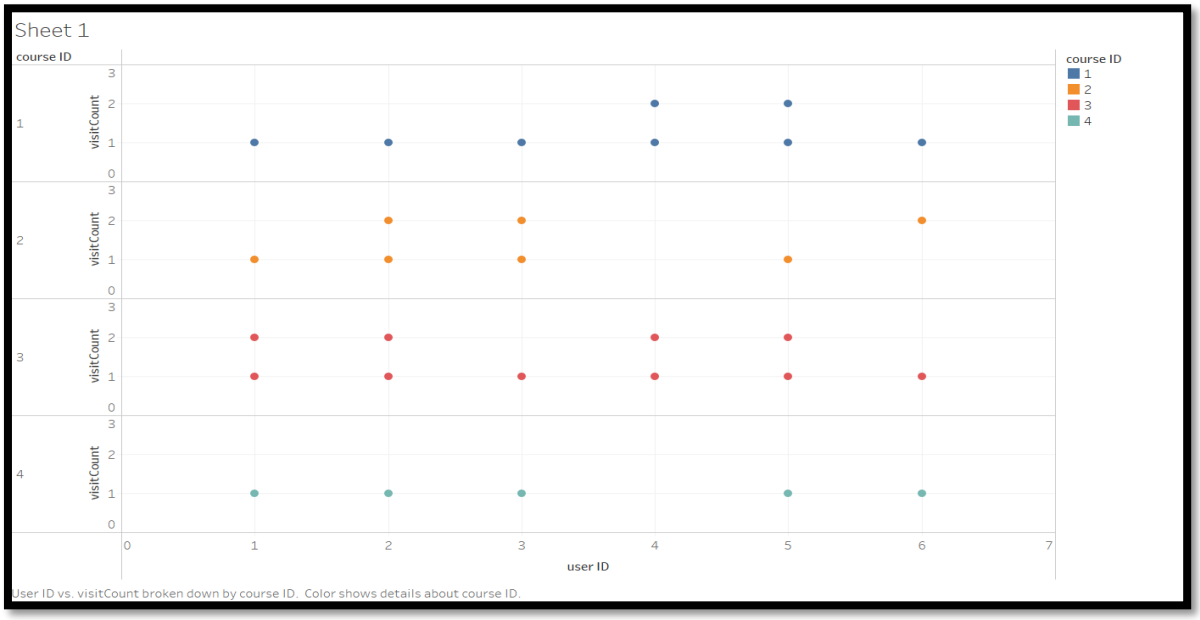
h. The user who visited a course the most times in a single day. (i.e., most frequent visited user in single day per course)

```sql
---8. Identify the user who visited each course the most times in a single day
WITH DailyVisitCounts AS (
    SELECT
        cv.courseID,
        cv.userID,
        cv.date,
        COUNT(cv.visitID) AS VisitCount,
        ROW_NUMBER() OVER (PARTITION BY cv.courseID, cv.date ORDER BY COUNT(cv.visitID) DESC) AS Rank
    FROM
        courseSiteVisit cv
    GROUP BY
        cv.courseID, cv.userID, cv.date
)
SELECT
    courseID,
    userID,
    date,
    VisitCount
FROM
    DailyVisitCounts
WHERE
    Rank = 1;
```

Output:

	courseID	userID	date	VisitCount
1	1	3	2023-05-14	1
2	1	4	2023-05-15	1
3	1	5	2023-05-17	1
4	1	3	2023-05-18	1
5	1	4	2023-05-20	1
6	1	6	2023-05-21	1
7	1	5	2023-05-23	1
8	1	1	2023-05-24	1
9	1	4	2023-05-25	1
10	1	5	2023-05-27	2
11	1	5	2023-05-28	1
12	1	2	2023-05-30	1
13	1	3	2023-05-31	1
14	1	4	2023-06-01	2
15	1	6	2023-06-02	1
16	1	6	2023-06-03	1
17	1	3	2023-06-06	1
18	1	1	2023-06-07	1
19	1	1	2023-06-08	1
20	2	2	2023-05-10	2

Visualization:



i. **Longest visit streak days per user per course (i.e., the maximum number of days in a row that a user has accessed or engaged with the course site)**

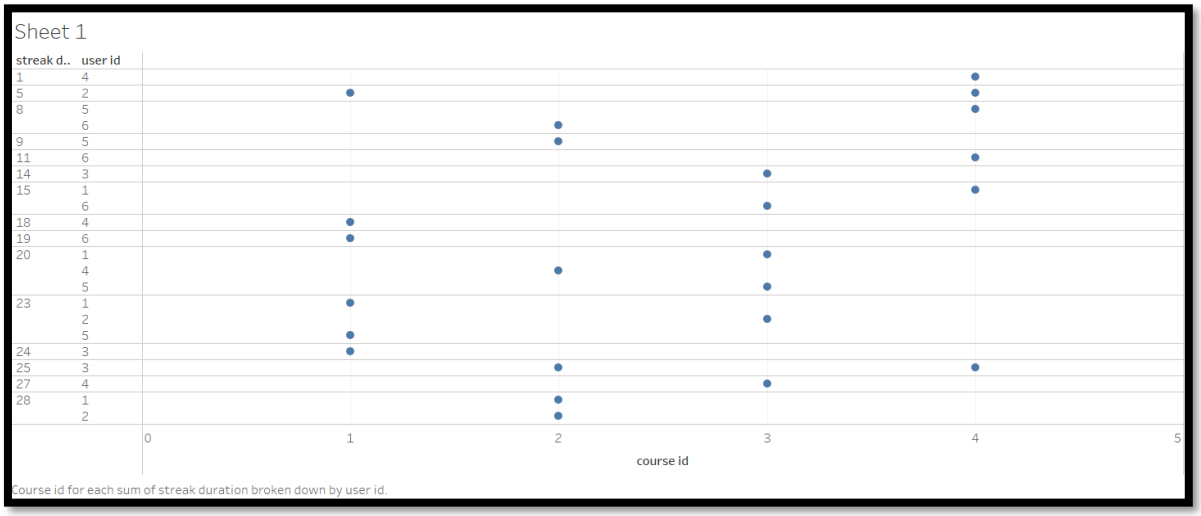
```sql
---9. Calculate the longest visit streak per user per course
WITH UserStreaks AS (
    SELECT
        cv.courseID,
        cv.userID,
        cv.date,
        ROW_NUMBER() OVER (PARTITION BY cv.courseID, cv.userID ORDER BY cv.date) -
        ROW_NUMBER() OVER (PARTITION BY cv.courseID, cv.userID ORDER BY cv.date) AS StreakGroup
    FROM
        courseSiteVisit cv
)

SELECT
    courseID,
    userID,
    MIN(date) AS StreakStartDate,
    MAX(date) AS StreakEndDate,
    DATEDIFF(day, MIN(date), MAX(date)) + 1 AS StreakDuration
FROM
    UserStreaks
GROUP BY
    courseID, userID, StreakGroup
ORDER BY
    courseID, userID, StreakStartDate;
```

Output:

	courseID	userID	StreakStartDate	StreakEndDate	StreakDuration
1	1	1	2023-05-17	2023-06-08	23
2	1	2	2023-05-30	2023-06-03	5
3	1	3	2023-05-14	2023-06-06	24
4	1	4	2023-05-15	2023-06-01	18
5	1	5	2023-05-17	2023-06-08	23
6	1	6	2023-05-20	2023-06-07	19
7	2	1	2023-05-11	2023-06-07	28
8	2	2	2023-05-10	2023-06-06	28
9	2	3	2023-05-15	2023-06-08	25
10	2	4	2023-05-12	2023-05-31	20
11	2	5	2023-05-31	2023-06-08	9
12	2	6	2023-05-29	2023-06-05	8
13	3	1	2023-05-18	2023-06-06	20
14	3	2	2023-05-12	2023-06-03	23
15	3	3	2023-05-26	2023-06-08	14
16	3	4	2023-05-10	2023-06-05	27
17	3	5	2023-05-12	2023-05-31	20
18	3	6	2023-05-13	2023-05-27	15
19	4	1	2023-05-22	2023-06-05	15
20	4	2	2023-05-29	2023-06-02	5

Visualization:



j. Longest gap between visit per user and number of days in single course

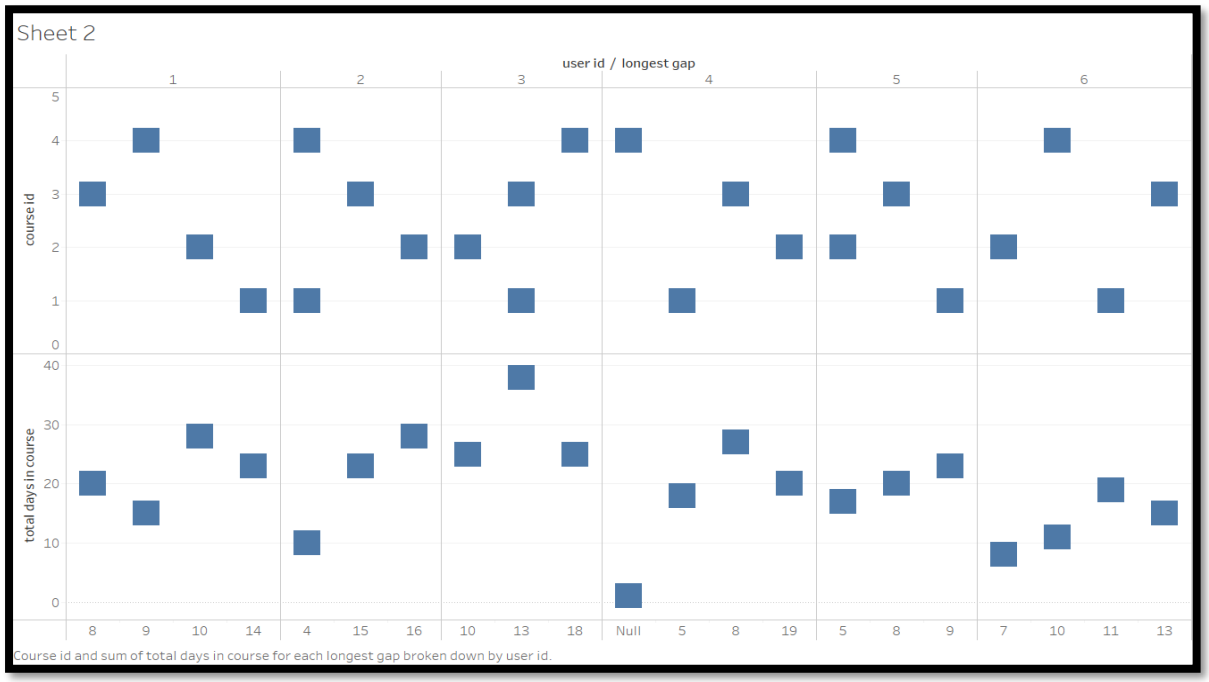
```sql
---10.  Identify the longest gap between visits per user and course
WITH UserVisitGaps AS (
    SELECT
        cv.courseID,
        cv.userID,
        cv.date,
        LAG(cv.date) OVER (PARTITION BY cv.courseID, cv.userID ORDER BY cv.date) AS PreviousVisitDate,
        DATEDIFF(day, LAG(cv.date) OVER (PARTITION BY cv.courseID, cv.userID ORDER BY cv.date), cv.date) AS GapDays
    FROM
        courseSiteVisit cv
)

SELECT
    courseID,
    userID,
    MAX(GapDays) AS LongestGap,
    DATEDIFF(day, MIN(date), MAX(date)) + 1 AS TotalDaysInCourse
FROM
    UserVisitGaps
GROUP BY
    courseID, userID;
```

Output:

	courseID	userID	LongestGap	TotalDaysInCourse
1	1	1	14	23
2	1	2	4	5
3	1	3	13	24
4	1	4	5	18
5	1	5	9	23
6	1	6	11	19
7	2	1	10	28
8	2	2	16	28
9	2	3	10	25
10	2	4	19	20
11	2	5	5	9
12	2	6	7	8
13	3	1	8	20
14	3	2	15	23
15	3	3	13	14
16	3	4	8	27
17	3	5	8	20
18	3	6	13	15
19	4	1	9	15
20	4	2	4	5

Visualization:



k. The user who visited the most courses within a short duration

```sql
---11. Identify the user who visited the most courses within a short duration
DECLARE @ShortDurationDays INT = 330;
WITH UserCourseCounts AS (
    SELECT
        cv.userID,
        COUNT(DISTINCT cv.courseID) AS CoursesVisited
    FROM
        courseSiteVisit cv
    WHERE
        cv.date >= DATEADD(day, -@ShortDurationDays, GETDATE())
    GROUP BY
        cv.userID
)

SELECT TOP 1
    userID,
    CoursesVisited
FROM
    UserCourseCounts
ORDER BY
    CoursesVisited DESC;
```

Output:

	userID	CoursesVisited
1	2	4