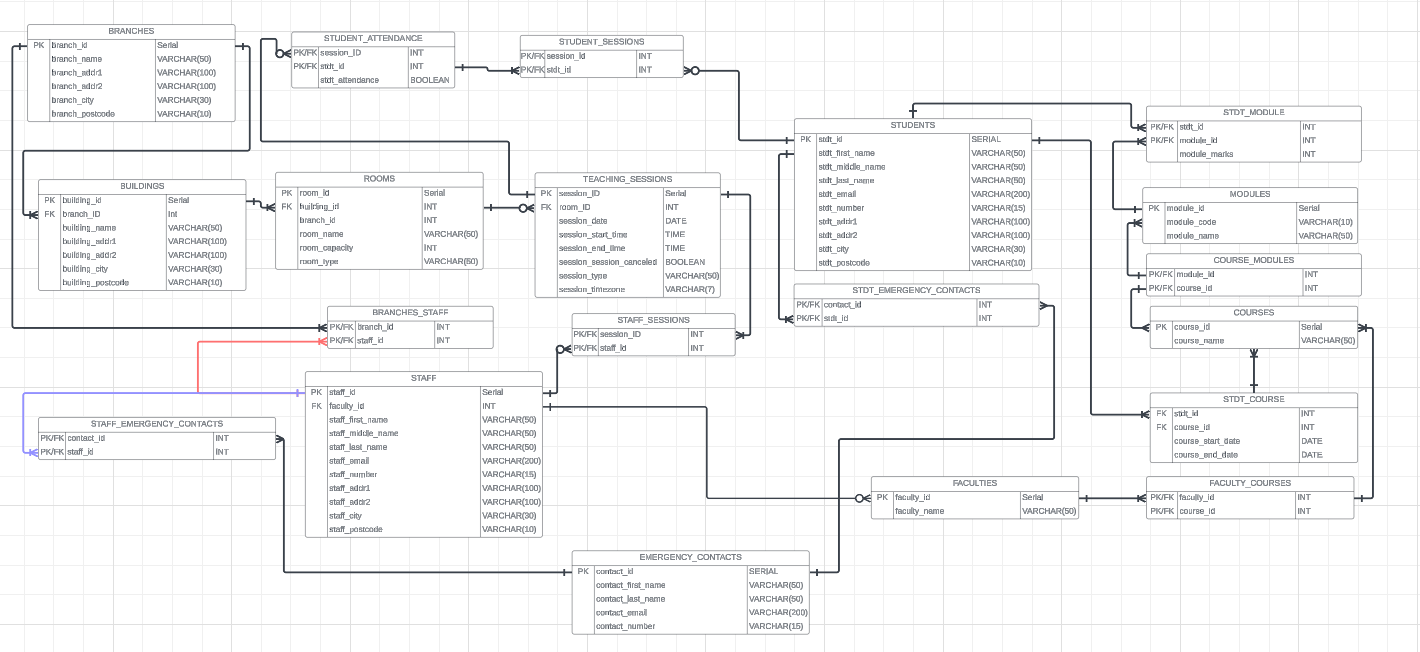
## 1.1 Contribution Statement

|  | **UP2212385** | **UP2204396** | **FUP2219449** |
| --- | --- | --- | --- |
| ERD Design |  |  |  |
| Assumptions |  |  |  |
| Data Dictionary |  |  |  |
| Coding (CREATE+INSERT) |  |  |  |
| Queries |  |  |  |
| Theoretical Aspects |  |  |  |
| Document Writing |  |  |  |
| Brainstorming |  |  |  |
| DDT |  |  |  |
| Percentage Allocation | **33.3%** | **33.3%** | **33.3%** |
| **Signature** |  |  |  |

## 1.2 Entity Relationship Diagram

****

[Link to the chart](https://lucid.app/lucidchart/e5a27ae4-fcad-4929-b758-a7038129ea37/edit?viewport_loc=-3362%2C-1319%2C10144%2C5175%2C0_0&invitationId=inv_bfa3b64b-7488-45a0-8a27-cc5a199b7983)

### **Assumptions**:

1) From their online tutoring roots, we assume that students can join physical sessions through a call, both locally and internationally.

2) We also assume that some sessions may be only online. So we will include a timezone in all sessions to accompany their teaching sessions. We will not be making a separate timezone table and that students will be able to calculate when they have their online sessions if their timezone differs from the session one.

3) One module can be on multiple courses so students from different courses can attend a lesson for that module.

4) One teacher will be under one faculty.

5) General work employees (e.g. Janitors, Receptionists, Marketing, Finance etc) will not be considered for this database and its purpose is for the education side so the only staff members in the STAFF table will be teachers/lecturers.

6) Rooms can be booked at any specific time (e.g. 13:15 -> 14:15, 8:18 -> 12:40).

We will impose no restrictions on room booking (e.g. time limit), any issues that come of this are assumed to be handled by the company internally (e.g. teacher books every room for a week).

7) Students' age range is 13+.

8) We assume that student attendance will want to be tracked.

9) We assume the system prevents students and teachers from being double-booked for overlapping sessions.

## 1.3 Data Dictionary

| STAFF | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| ***Attribute\_Name*** | ***KEY*** | ***INDEX*** | ***Data Type & Size*** | ***Domains***  ***&***  ***Constraints*** | ***FK Reference*** | ***Description*** |
| staff\_id | PK |  | SERIAL |  |  |  |
| branch\_id | FK |  | INT |  | branches.branch\_id |  |
| faculty\_id | FK |  | INT |  | faculties.faculty\_id |  |
| staff\_first \_name |  |  | VARCHAR(50) | NOT NULL |  |  |
| staff\_middle\_name |  |  | VARCHAR(50) |  |  |  |
| staff\_last\_name |  | Y | VARCHAR(50) | NOT NULL |  |  |
| staff\_email |  |  | VARCHAR(200) | NOT NULL |  |  |
| staff\_number |  |  | VARCHAR(15) | NOT NULL |  | Staff phone number |
| staff\_addr1 |  |  | VARCHAR(100) | NOT NULL |  |  |
| staff\_addr2 |  |  | VARCHAR(100) |  |  |  |
| staff\_city |  |  | VARCHAR(30) | NOT NULL |  |  |
| staff\_postcode |  |  | VARCHAR(10) | NOT NULL |  |  |

| FACULTIES | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| ***Attribute\_Name*** | ***KEY*** | ***INDEX*** | ***Data Type & Size*** | ***Domains***  ***&***  ***Constraints*** | ***FK Reference*** | ***Description*** |
| faculty\_id | PK |  | SERIAL |  |  |  |
| faculty\_name |  |  | VARCHAR(50) | NOT NULL, UNIQUE |  |  |

| FACULTY\_COURSES | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| ***Attribute\_Name*** | ***KEY*** | ***INDEX*** | ***Data Type & Size*** | ***Domains***  ***&***  ***Constraints*** | ***FK Reference*** | ***Description*** |
| faculty\_id | PK/FK |  | INT |  | faculties.faculty\_id |  |
| course\_id | PK/FK |  | INT |  | courses.course\_id |  |

| EMERGENCY\_CONTACTS | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| ***Attribute\_Name*** | ***KEY*** | ***INDEX*** | ***Data Type & Size*** | ***Domains***  ***&***  ***Constraints*** | ***FK Reference*** | ***Description*** |
| contact\_id | PK |  | SERIAL |  |  |  |
| contact\_first\_name |  |  | VARCHAR(50) | NOT NULL |  |  |
| contact\_last\_name |  |  | VARCHAR(50) | NOT NULL |  |  |
| contact\_email |  |  | VARCHAR(200) | NOT NULL |  |  |
| contact\_number |  |  | VARCHAR(15) | NOT NULL |  |  |

| STAFF\_EMERGENCY\_CONTACTS | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| ***Attribute\_Name*** | ***KEY*** | ***INDEX*** | ***Data Type & Size*** | ***Domains***  ***&***  ***Constraints*** | ***FK Reference*** | ***Description*** |
| staff\_id | PK/FK |  | INT |  | staff.staff\_id |  |
| contact\_id | PK/FK |  | INT |  | emergency\_contacts.contact\_id |  |

| TEACHING\_SESSIONS | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| ***Attribute\_Name*** | ***KEY*** | ***INDEX*** | ***Data Type & Size*** | ***Domains***  ***&***  ***Constraints*** | ***FK Reference*** | ***Description*** |
| sessions\_ID | PK |  | SERIAL |  |  |  |
| room\_ID | FK |  | INT |  | rooms.room\_id |  |
| sessions\_date |  |  | DATE | NOT NULL |  |  |
| sessions\_start\_time |  |  | TIME | NOT NULL |  |  |
| sessions\_end\_time |  |  | TIME | NOT NULL |  |  |
| session\_session\_cancelled |  |  | BOOLEAN | NOT NULL |  |  |
| session\_type |  |  | VARCHAR(50) | NOT NULL |  |  |
| sessions\_timezone |  |  | VARCHAR(7) | NOT NULL |  | Timezone used for session time (some students may be international and on virtual sessions) |

| STAFF\_SESSIONS | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| ***Attribute\_Name*** | ***KEY*** | ***INDEX*** | ***Data Type & Size*** | ***Domains***  ***&***  ***Constraints*** | ***FK Reference*** | ***Description*** |
| sessions\_ID | PK/FK |  | INT |  | teaching\_sessions.sessions\_id |  |
| staff\_id | PK/FK |  | INT |  | staff.staff\_id |  |

| STUDENT\_SESSIONS | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| ***Attribute\_Name*** | ***KEY*** | ***INDEX*** | ***Data Type & Size*** | ***Domains***  ***&***  ***Constraints*** | ***FK Reference*** | ***Description*** |
| session\_id | PK/FK |  | INT | NOT NULL | teaching\_sessions.session\_ID |  |
| stdt\_id | PK/FK |  | INT | NOT NULL | students.stdt\_ID |  |
| stdt\_attendence |  |  | BOOLEAN | NOT NULL |  | States whether or not a student has attended the specified teaching session |

| ROOMS | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| ***Attribute\_Name*** | ***KEY*** | ***INDEX*** | ***Data Type & Size*** | ***Domains***  ***&***  ***Constraints*** | ***FK Reference*** | ***Description*** |
| room\_id | PK |  | SERIAL |  |  |  |
| building\_id | FK |  | INT |  | buildings.building\_id |  |
| branch\_id |  |  | INT |  |  |  |
| room\_name |  |  | VARCHAR(50) | NOT NULL |  |  |
| room\_type |  |  | VARCHAR(50) | NOT NULL |  | The type of room e.g classroom, computer lab |
| room\_capacity |  |  | INT | NOT NULL |  | How many students can fit in the room. |

| BUILDINGS | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| ***Attribute\_Name*** | ***KEY*** | ***INDEX*** | ***Data Type & Size*** | ***Domains***  ***&***  ***Constraints*** | ***FK Reference*** | ***Description*** |
| building\_ID | PK |  | SERIAL |  |  |  |
| branch\_ID | FK |  | INT |  | branches.branch\_id |  |
| building\_name |  |  | VARCHAR(50) | NOT NULL |  |  |
| building\_addr1 |  |  | VARCHAR(100) | NOT NULL |  |  |
| building\_addr2 |  |  | VARCHAR(100) |  |  |  |
| building\_city |  |  | VARCHAR(30) | NOT NULL |  |  |
| building\_postcode |  |  | VARCHAR(10) | NOT NULL |  |  |

| BRANCHES | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| ***Attribute\_Name*** | ***KEY*** | ***INDEX*** | ***Data Type & Size*** | ***Domains***  ***&***  ***Constraints*** | ***FK Reference*** | ***Description*** |
| branch\_ID | PK |  | SERIAL |  |  |  |
| branch\_name |  |  | VARCHAR(50) | NOT NULL, UNIQUE |  |  |
| branch\_addr1 |  |  | VARCHAR(100) | NOT NULL |  |  |
| branch\_addr2 |  |  | VARCHAR(100) |  |  |  |
| branch\_city |  |  | VARCHAR(30) | NOT NULL |  |  |
| branch\_postcode |  |  | VARCHAR(10) | NOT NULL |  |  |

| BRANCHES\_STAFF | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| ***Attribute\_Name*** | ***KEY*** | ***INDEX*** | ***Data Type & Size*** | ***Domains***  ***&***  ***Constraints*** | ***FK Reference*** | ***Description*** |
| branch\_id | PK/FK |  | INT |  | branches.branch\_id |  |
| staff\_id | PK/FK |  | INT |  | staff.staff\_id |  |

| STUDENTS | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| ***Attribute\_Name*** | ***KEY*** | ***INDEX*** | ***Data Type & Size*** | ***Domains***  ***&***  ***Constraints*** | ***FK Reference*** | ***Description*** |
| stdt\_id | PK |  | SERIAL |  |  |  |
| stdt\_first\_name |  |  | VARCHAR(50) | NOT NULL |  |  |
| stdt\_middle\_name |  |  | VARCHAR(50) |  |  |  |
| stdt\_last\_name |  | Y | VARCHAR(50) | NOT NULL |  |  |
| stdt\_email |  |  | VARCHAR(200) | NOT NULL |  |  |
| stdt\_number |  |  | VARCHAR(15) | NOT NULL |  |  |
| stdt\_addr1 |  |  | VARCHAR(100) | NOT NULL |  |  |
| stdt\_addr2 |  |  | VARCHAR(100) |  |  |  |
| stdt\_city |  |  | VARCHAR(30) | NOT NULL |  |  |
| stdt\_postcode |  |  | VARCHAR(10) | NOT NULL |  |  |

| STDT\_EMERGENCY\_CONTACTS | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| ***Attribute\_Name*** | ***KEY*** | ***INDEX*** | ***Data Type & Size*** | ***Domains***  ***&***  ***Constraints*** | ***FK Reference*** | ***Description*** |
| contact\_id | PK/FK |  | INT |  | emergency\_contacts.contact\_id |  |
| stdt\_id | PK/FK |  | INT |  | students.stdt\_id |  |

| STDT\_MODULE | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| ***Attribute\_Name*** | ***KEY*** | ***INDEX*** | ***Data Type & Size*** | ***Domains***  ***&***  ***Constraints*** | ***FK Reference*** | ***Description*** |
| stdt\_id | PK/FK |  | INT |  | students.stdt\_id |  |
| module\_id | PK/FK |  | INT |  | modules.module\_id |  |
| module\_marks |  |  | INT |  |  |  |

| MODULES | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| ***Attribute\_Name*** | ***KEY*** | ***INDEX*** | ***Data Type & Size*** | ***Domains***  ***&***  ***Constraints*** | ***FK Reference*** | ***Description*** |
| module\_id | PK | Y | SERIAL |  |  |  |
| module\_code |  |  | VARCHAR(10) | NOT NULL, UNIQUE |  |  |
| module\_name |  |  | VARCHAR(50) | NOT NULL |  |  |

| COURSE\_MODULES | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| ***Attribute\_Name*** | ***KEY*** | ***INDEX*** | ***Data Type & Size*** | ***Domains***  ***&***  ***Constraints*** | ***FK Reference*** | ***Description*** |
| module\_id | PK/FK |  | INT |  | modules.module\_id |  |
| course\_id | PK/FK |  | INT |  | courses.course\_id |  |

| COURSES | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| ***Attribute\_Name*** | ***KEY*** | ***INDEX*** | ***Data Type & Size*** | ***Domains***  ***&***  ***Constraints*** | ***FK Reference*** | ***Description*** |
| course\_id | PK |  | SERIAL |  |  |  |
| course\_name |  |  | VARCHAR(50) | NOT NULL, UNIQUE |  |  |

| STDT\_COURSE | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| ***Attribute\_Name*** | ***KEY*** | ***INDEX*** | ***Data Type & Size*** | ***Domains***  ***&***  ***Constraints*** | ***FK Reference*** | ***Description*** |
| stdt\_id | PK/FK |  | INT |  | students.stdt\_id |  |
| course\_id | PK/FK |  | INT |  | course.course\_id |  |
| course\_start\_date |  |  | DATE | NOT NULL |  |  |
| course\_end\_date |  |  | DATE | NOT NULL |  |  |

| STUDENT\_ATTENDANCE | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| ***Attribute\_Name*** | ***KEY*** | ***INDEX*** | ***Data Type & Size*** | ***Domains***  ***&***  ***Constraints*** | ***FK Reference*** | ***Description*** |
| session\_ID | PK/FK |  | INT |  | student\_sessions.session\_id |  |
| stdt\_id | PK/FK |  | INT |  | student\_sessions.stdt\_id |  |
| stdt\_attendance |  |  | BOOLEAN | NOT NULL |  |  |

## 2.2 5 Business Related Queries

Query 1:

This query retrieves a list of faculties along with the courses they manage, including courses with no faculty assigned.

SELECT

F.faculty\_name AS "Faculty Name",

C.course\_name AS "Course Name",

COUNT(CM.module\_id) AS "Number of Modules"

FROM

FACULTIES F

JOIN

FACULTY\_COURSES FC ON F.faculty\_id = FC.faculty\_id

JOIN

COURSES C ON FC.course\_id = C.course\_id

LEFT JOIN

COURSE\_MODULES CM ON C.course\_id = CM.course\_id

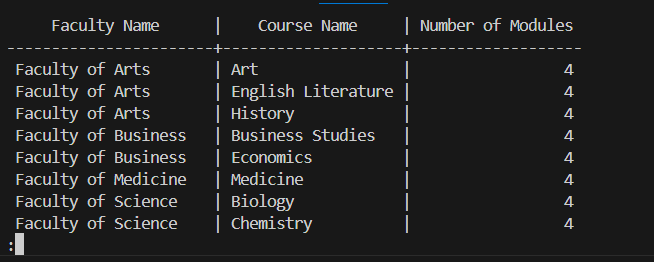
GROUP BY

F.faculty\_name, C.course\_name

ORDER BY

COUNT(CM.module\_id) DESC, F.faculty\_name, C.course\_name;

Output:



Query 2:

This query lists staff members along with their emergency contacts, showing staff with no contacts(if none added) as well.

SELECT

S.staff\_first\_name || ' ' || S.staff\_last\_name AS "Staff Name",

EC.contact\_first\_name || ' ' || EC.contact\_last\_name AS "Emergency Contact",

EC.contact\_email AS "Emergency Contact Email",

EC.contact\_number AS "Emergency Contact Number"

FROM

STAFF S

LEFT JOIN

STAFF\_EMERGENCY\_CONTACTS SEC ON S.staff\_id = SEC.staff\_id

LEFT JOIN

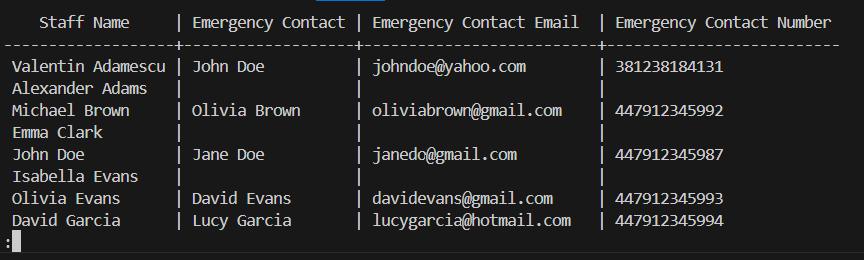
EMERGENCY\_CONTACTS EC ON SEC.contact\_id = EC.contact\_id

ORDER BY

S.staff\_last\_name, S.staff\_first\_name;

S.staff\_last\_name, S.staff\_first\_name;

Output:



Query 3:

Retrieves information about teachers, including their names, email addresses, and the number of sessions they have taught. Highlights the tutors engagement in each teaching session.

SELECT

S.staff\_first\_name || ' ' || S.staff\_last\_name AS "Teacher Name",

S.staff\_email AS "Email",

COUNT(SF.session\_ID) AS "Sessions Taught",

B.branch\_name AS "Branch Name",

F.faculty\_name AS "Faculty Name"

FROM

STAFF S

LEFT JOIN

STAFF\_SESSIONS SF ON S.staff\_id = SF.staff\_id

LEFT JOIN

BRANCHES B ON S.branch\_id = B.branch\_id

LEFT JOIN

FACULTIES F ON S.faculty\_id = F.faculty\_id

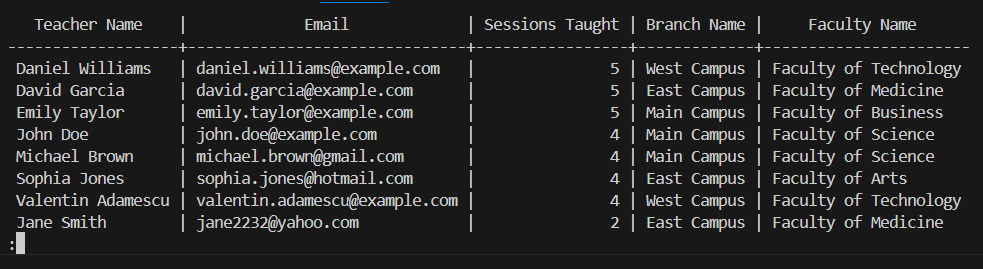
GROUP BY

S.staff\_first\_name, S.staff\_last\_name, S.staff\_email, B.branch\_name, F.faculty\_name

ORDER BY

"Sessions Taught" DESC, "Teacher Name", "Branch Name", "Faculty Name";

Output:



Query 4:

Student Attendance Summary, This query provides a summary of attendance for each student, showing the total sessions and attended sessions.

SELECT

S.stdt\_first\_name || ' ' || S.stdt\_last\_name AS "Student Name",

COUNT(DISTINCT TS.session\_id) AS "Total Sessions",

SUM(CASE WHEN SA.stdt\_attendance = TRUE THEN 1 ELSE 0 END) AS "Attended Sessions",

B.branch\_name AS "Branch Name"

FROM

STUDENTS S

LEFT JOIN

STUDENT\_SESSIONS ST ON S.stdt\_id = ST.stdt\_id

LEFT JOIN

TEACHING\_SESSIONS TS ON ST.session\_ID = TS.session\_id

LEFT JOIN

STUDENT\_ATTENDANCE SA ON ST.session\_ID = SA.session\_id AND S.stdt\_id = SA.stdt\_id

LEFT JOIN

BRANCHES B ON S.branch\_id = B.branch\_id

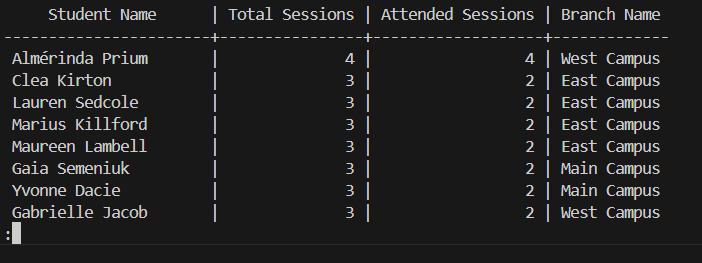
GROUP BY

S.stdt\_first\_name, S.stdt\_last\_name, B.branch\_name

ORDER BY

"Total Sessions" DESC, "Attended Sessions" DESC, "Branch Name" ASC, "Student Name" ASC;

Output:



Query 5:

List of Students who missed their sessions. This query retrieves a list of students along with the number of sessions they missed. It shows the student name and the count of missed sessions (and their emergency contacts if they have any)

SELECT

S.stdt\_first\_name || ' ' || S.stdt\_last\_name AS "Student Name",

S.stdt\_email AS "Student Email",

S.stdt\_number AS "Student Number",

EC.contact\_first\_name || ' ' || EC.contact\_last\_name AS "Emergency Contact",

EC.contact\_email AS "Emergency Contact Email",

EC.contact\_number AS "Emergency Contact Number",

COUNT(SA.session\_id) AS "Missed Sessions"

FROM

STUDENTS S

LEFT JOIN

STUDENT\_SESSIONS ST ON S.stdt\_id = ST.stdt\_id

LEFT JOIN

STUDENT\_ATTENDANCE SA ON ST.session\_ID = SA.session\_id AND SA.stdt\_attendance = FALSE

LEFT JOIN

STDT\_EMERGENCY\_CONTACTS SEC ON S.stdt\_id = SEC.stdt\_id

LEFT JOIN

EMERGENCY\_CONTACTS EC ON SEC.contact\_id = EC.contact\_id

GROUP BY

S.stdt\_first\_name, S.stdt\_last\_name, S.stdt\_email, S.stdt\_number, EC.contact\_first\_name, EC.contact\_last\_name, EC.contact\_email, EC.contact\_number

HAVING

COUNT(SA.session\_id) > 0

ORDER BY

"Missed Sessions" DESC;

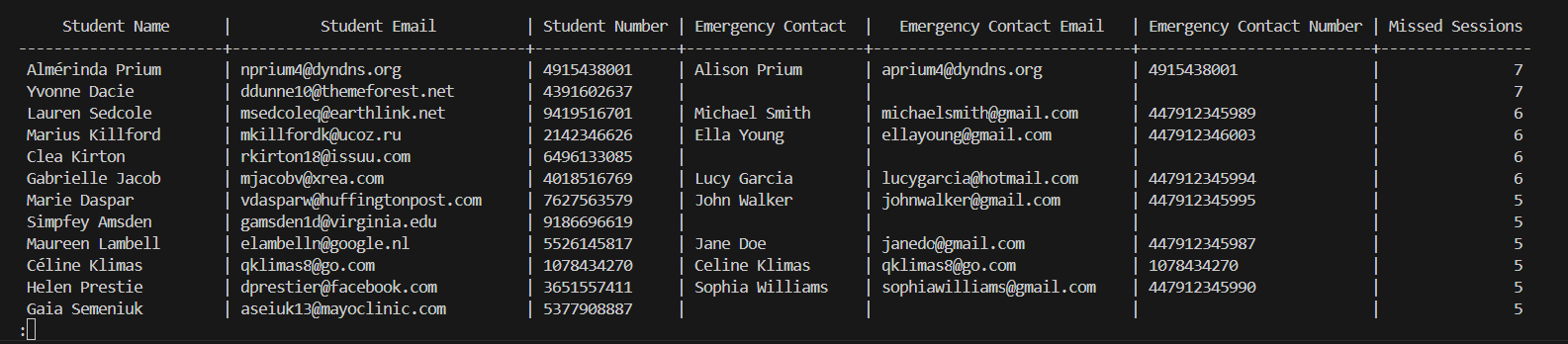
HAVING

COUNT(SA.session\_id) > 0

ORDER BY

"Missed Sessions" DESC;

Output:



**(Theoretical Aspects, Security and Optimisation)**

## 

## 3.1 Security

For our security, we implemented the following roles to our database:

* Student
* Staff
* Database admin

Starting with the database admin, we have decided to give all privileges to this role. This is because it is the role that is there to help implement and administer anything that relates to the database. So therefore, giving this role access to all the tools available to complete their job makes the most sense, therefore keeping the business running as smoothly as possible.

For the staff role, we have given SELECT access on ROOMS, MODULES, STAFF, STDT\_MODULE, COURSE, STUDENT\_ATTENDANCE, as well as select, insert, update and delete on both TEACHING\_SESSIONS and STUDENT\_SESSIONS. The reason we have decided to do this is because it gives them the ability to access some crucial queries, such as Student Attendance Summary, which provides a summary of attendance for each student. Additionally, giving the privileges to TEACHING\_SESSIONS and STUDENT\_SESSIONS allows them to update crucial information about these tables, in this case, to book teaching sessions and assign them to the relevant students. Because they are staff, there is a certain level of trust established that allows them to be given these privileges. In conclusion, giving the staff these privileges allows them to help perform tasks that help keep the case study running smoothly, therefore helping to keep the business executing efficiently.

In contrast, for the student role, we have only granted select on the view student\_attendance\_summary\_view and on the tables STDT\_MODULE and STDT\_COURSE. By only granting select on a view, it means that the student can only access the relevant data from the tables associated with that view, meaning they can’t access any data they shouldn’t. Giving them select access to stdt\_module and stdt\_course allows them to see relevant information such as their module marks or their course start or end date. Unlike the staff role, students have much less trust associated with them, so therefore their roles must reflect this by being much more restrictive; we don’t want to give them access to data that is not relevant to them, as it could potentially be used maliciously. However, there is still some data that they need to access, as it helps with the progression of their studying, therefore helping the business to further run efficiently.

REVOKE ALL ON DATABASE DB FROM public;

CREATE ROLE database\_admin;

CREATE ROLE staff;

CREATE ROLE student;

GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO database\_admin;

GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO database\_admin;

GRANT ALL PRIVILEGES ON ALL FUNCTIONS IN SCHEMA public TO database\_admin;

GRANT SELECT, INSERT, UPDATE, DELETE ON TEACHING\_SESSIONS TO staff;

GRANT SELECT, INSERT, UPDATE, DELETE ON STUDENT\_SESSIONS TO staff;

GRANT SELECT ON ROOMS TO staff;

GRANT SELECT ON MODULES TO staff;

GRANT SELECT ON STAFF TO staff;

GRANT SELECT ON STDT\_MODULE TO staff;

GRANT SELECT ON STDT\_COURSE TO staff;

GRANT SELECT ON STUDENT\_ATTENDANCE TO staff;

GRANT SELECT ON branches TO staff;

GRANT SELECT ON student\_attendance\_summary\_view TO student;

GRANT SELECT ON STDT\_MODULE TO student;

GRANT SELECT ON STDT\_COURSE TO student;

## 3.2 Optimisation and Transaction Analysis:

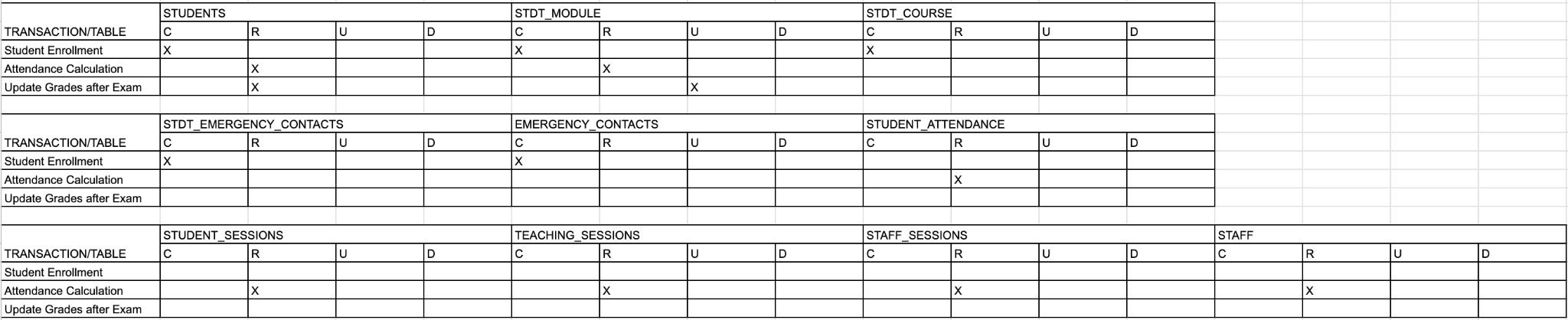
For our three performance-sensitive transactions, we have chosen the following:

* Student enrollment - automatically inserts the student in to all required areas of the database
* Attendance calculation - will calculate the attendance of each student, which can be either for each module, or their entire attendance over the whole course
* Grades updater after exams - would allow staff to import a large quantity of marks, which will therefore update each individual student.

There are many reasons why we feel like these are the most performance-sensitive, starting with frequency. For transactions like attendance calculation, it would be used multiple times a day for each student enrolled, meaning the performance for a query like this is crucial. Additionally, transactions like updating grades or student enrollment would be used a lot within certain time periods, i.e. exam periods or the upcoming months before the first term starts, therefore also making it very performance sensitive. As well as this, the attendance calculation transaction also has a high level of complexity due to the amount of joins it uses (as can be seen by our transaction analysis matrix), which also makes it performance sensitive. Finally, the thing that all 3 of these queries have in common is that they are very important to the business of the case study. They all are used for crucial operations of the business, meaning that their high impact also makes them performance sensitive, as if they weren’t properly optimised or stopped working, it could cause issues for the business.

In order to make these transactions perform as well as possible, we would implement a number of different optimisation techniques, one of these being indexing. In our database, we have already implemented indexes on columns such as student\_id, course\_id and session\_id. For transactions such as attendance calculation and grades updater, having indexes on student\_id is useful, as these are frequently used by these transactions. Additionally, having the index on session\_id is also helpful in the attendance calculation transaction. Another optimisation technique that could be used here to further improve the performance of these transactions is denormalization. Whilst all the tables in our database are already normalised to the 3rd form, transactions like our attendance calculation can benefit from this. This is because denormalisation helps to reduce the number of joins required, therefore reducing the complexity of transaction and allowing it to perform more efficiently. With the attendance calculation, denormalizing tables such as students and stdt\_module or student\_attendance and student\_sessions together can help achieve this. Finally, we could also use VIEWS with these transactions to improve their performance. With attendance calculation, once calculated after the most recent update, a view can be created on it so that the transaction does not have to be performed each time the attendance is trying to be checked. This helps benefit all transactions, as processing power can be used elsewhere, since the table is stored on the database rather than being processed each time.

**Transaction Analysis Matrix**



## 3.3 Professional, Legal and Ethical Issues

In developing the database system for Stellar Education Solutions (SES), we will prioritize adherence to relevant legal frameworks, particularly the General Data Protection Regulation (GDPR) and the UK Data Protection Act. Our approach will ensure that we only store the minimum amount of information necessary to provide our tutoring services. We will not track, store, or request any unnecessary data for the database, thereby minimizing potential privacy risks.

Given that students may be as young as 13 years old, it is crucial to obtain explicit user consent from both students and guardians where applicable. This is essential not only for legal reasons but also to uphold ethical standards in our operations. We will use clear and straightforward consent forms that outline how data will be used, stored, and protected, ensuring all parties are fully informed.

As for Roles and Privileges, our database will include a roles and privileges system to ensure that only authorized personnel can access sensitive data. This means that specific roles, such as branch managers and administrators, will have varying levels of access rights, while teachers may have restricted access to only the information necessary for their teaching responsibilities. This separation of duties is vital for maintaining data confidentiality and integrity.

Lastly, the encryption of sensitive information. To further safeguard sensitive information, we will implement encryption protocols for data at rest and in transit. Personal data, such as student names, contact information, and performance metrics, will be encrypted to protect against unauthorized access and data breaches. This encryption will comply with industry standards and best practices, ensuring that even in the event of a data breach, the information remains secure and unreadable without the appropriate decryption keys.

By adopting these professional, legal, and ethical measures, we aim to create a secure and trustworthy database system for SES that aligns with the principles of data protection and respects the privacy of our users.

## 3.4 Database Development Tracker

| **Date** | **Task Description** | **Member ID** | **Task Details** | **Time** | **Signature** |
| --- | --- | --- | --- | --- | --- |
| 03/10/2024 | Initial meeting, Brainstorming and discussion | UP2219449  UP2212385  UP2204396 | We have met initially to discuss the CW details and organise future meetings. We have completed the first ERD and done the assumptions | 2 hours |  |
| 10/10/2024 | Progression on the coursework. ERD, DATA DICTIONARY | UP2219449  UP2212385  UP2204396 | We have finished and built on the ERD and made progress on the Data Dictionary | 2 hours |  |
| 24/10/2025 | Finishing the bulk of the coursework and building on previous work (finding improvements / errors) | UP2219449  UP2212385  UP2204396 | Finished the Data Dictionary, Made the SQL Code and made some inserts for the coursework, brainstormed any potential improvements to our ERD as well as errors that need to be changed.  Made some queries | 4.5 hours |  |
| 15/11/2024 | Finalising Queries and fixing errors on SQL code | UP2219449  UP2212385  UP2204396 | Finished the SQL code and made 5 complex and detailed Queries for your SQL code. Fixing code errors and adding inserts. | 2 hours |  |
| 21/11/2024 | Re working project and adding more necessary changes | UP2219449  UP2212385  UP2204396 | Changing our ERD to include student sessions and buildings. Fixing data dictionaries. Fixing sql code. Fixing queries. Doing Roles, Security, Write-up and optimisation | 3  hours |  |
| 07/12/24 | Working remotely | UP2219449  UP2212385  UP2204396 | Working remotely to do transaction analysis and fix databases. Making Views | 2hrs |  |
| 14/12/24 | Finalising work | UP2219449  UP2212385  UP2204396 | Finishing off project.  Working on the final aspects of the Database. Agreeing on security and optimisation. Writing the essay. Correcting any final errors and syntax or mistakes. Agreeing on a suitable final submission | 3hrs |  |
| 17/12/24 | Submission of work | UP2219449  UP2212385  UP2204396 | Proof reading to ensure we all agree and are satisfied with the quality of work done and equal contribution. Final additions and minor changes to Essays written under Optimisation and Security | 3hrs |  |