Project 1: Student Assessment System using Open University Learning Analytics dataset

Anant Jain (2019MCS2557) Pritesh Kumar Srivastava (2019MCS2658)

Due date: March 2, 2020, 11:55pm IST

1 Project Description

Our project involves analysis of *Open University Learning Analytics dataset*. We developed a web-based project for analysis and we are calling it *Student Assessment System*. This focuses on the collection and analysis of students and teachers data to improve their learning experience by providing informed guidance and to optimise learning materials. In this datasets, we have data about the high education , regions , age band , imb band etc of the students. Students who are studying the course , have provided them with *Virtual Learning Environment*. It contain information about logs of their interactions with the VLE represented by daily summaries of student clicks.

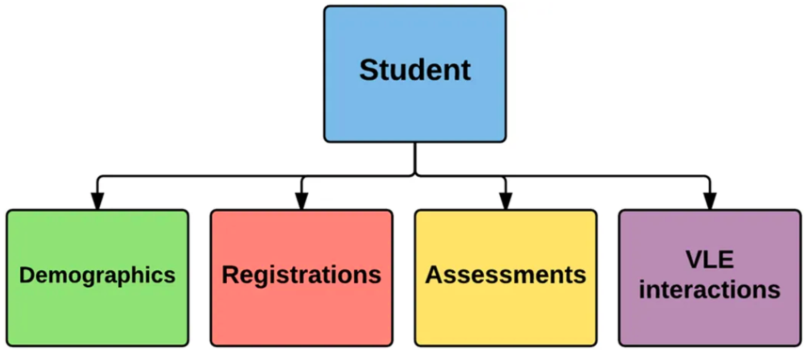

Figure 1: Overall dataset structure

The Open University is one of the largest distance learning universities where students are registered in different programs. Teaching materials and other content are delivered to students via the VLE. Students' interactions with the educational materials are recorded and stored in database for further analysis.

The idea behind the project is to analyse the performance of student who have registered in courses conducted by Virtual Learning Environment. Student can register themselves using Registration page and can see their assessments for the registered courses.

Teacher can see the student performances and analyze by using different options and parameters at online medium.

After selection of Dataset, our first task is to clean and customized it as per our project requirements. For this we have created some more tables and populated it with meaningful data. We have also created constraints(primary keys, foreign key and check) and sequences. For the speedup of queries we have created indexes and materialized views in database.

2 Data Sources and statistics

The *Open University Learning Analytics dataset* were picked up from the link:-
`https://analyse.kmi.open.ac.uk/open_dataset#data`.

Open University Learning Analytics Dataset (OULAD) contains data about courses, students and their interactions with Virtual Learning Environment (VLE) for seven selected courses (called modules). Presentations of courses start in February and October - they are marked by "B" and "J" respectively. Data set is available as a set of separate CSV files (comma separated values, each value is within quotation marks and the first line represents column names). Each file contains one 'database' table. Some tables are connected using unique identifiers (columns).

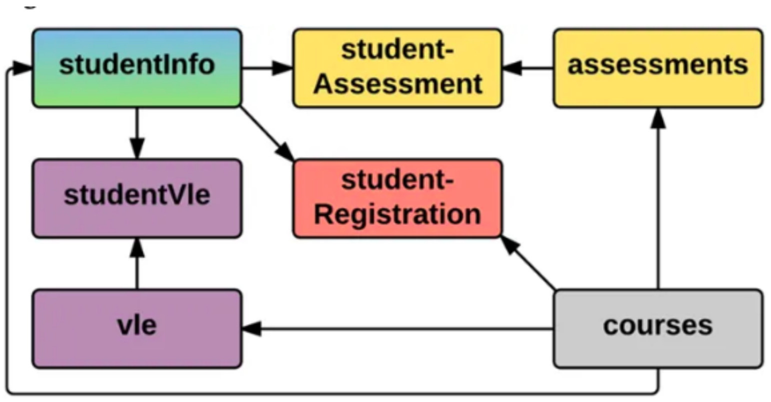

Figure 2: Detail dataset structure

Figure 2 shows the detailed structure of the dataset. Table *studentInfo* can be linked to *studentAssessment*, *studentVle* and *studentRegistration* tables using column *id_student*. Table *courses* links to the *assessments*, *studentRegistration*, *vle* and *studentInfo* using identifier columns *code_module* and *code_presentation*. Finally assessments table links to *studentAssessment* using *id_assessment* and *vle* to *studentVle* using *id_site*.

We have created tables and copied CSV data using sql copy command. We have also noted the load time of CSV data. As per our project requirements, some more tables are created and populated by data script.

The table 1 lists all Entities, Attributes, Number of tuples and Load Time.

Entity Name	**Attributes**	**No. of Tuples**	**Load Time(ms)**
courses	code_module, code_presentation,length	22	43.534 ms
assessments	code_module, code_presentation, id_assessment, assessment_type, date, weight	206	45.061 ms
studentinfo	code_module, code_presentation, id_student, gender, region, highest_education, imd_band, age_band, num_of_prev_attempts, studied_credits, disability, final_result	32593	470.076 ms
studentassessment	id_assessment, id_student, date_submitted, is_banked, score	173912	1287.160 ms
studentregistration	code_module, code_presentation, id_student, date_registration, date_unregistration	32593	167.549 ms
vle	id_site, code_module, code_presentation, activity_type, week_from, week_to	6364	58.973 ms
studentvle	id_studentvle, code_module, code_presentation, id_student, id_site, date, sum_click	10655280	203727.623 ms
login_info	login_id, password, login_role, person_id	28791	303.74 ms
teacher	id_teacher, name, profile, qualification, about, expertise, email, phone_no	6	11.94 ms
teacher_course	id_teacher, code_module, code_presentation, subject_name	22	14.16 ms
subject_name	code_module, subject_name	7	9.19 ms
course_commencement_date	code_presentation, commencement_date	4	33.02 ms
studentinfo	id_student, name, gender, region	28785	433.16 ms

Table 1: Entities, Attributes, No. of Tuples and Load Time

3 Functionality and Working

3.1 User's View of the System

3.1.1 Registration

This module enable student to create login credentials. Students are required valid email id for registration. The registration information will be stored in login_info table.

3.1.2 Login

If the users are already registered then they can login in the system using email id and password. When the password correct only then user can login in the system.

3.1.3 Teacher

This module can be accessed by the user who has logged in the system as 'Teacher' role. Teachers can see the current courses taught by them, students enrolled for that course and student's assessments. Teachers can also see their profile information which will be stored in 'teacher' and 'teacher_courses' tables. Also various useful information for teachers regarding courses, student and assessments available at a common platform. These information will be retrieved using User Defined Functions (UDFs).

3.1.4 Student

Student can access this module by logging in system as 'Student' role. They can see different available courses, teachers information, VLE (Virtual Learning Environment) and result/credit of assessments. Student's information will be maintained in following tables:

- StudentInfo
- StudentRegistration
- StudentAssessment
- StudentVLE

3.1.5 Admin

Admin has super user privilege and responsible of overall management of technical aspects of the system. Admin can see and edit information about courses, teachers, students, assessments and online material information. There will be common platform where admin can view and manage the system.

3.2 Internal Implementation

3.2.1 New Tables

In order to achieve consistency and completeness, we have created following new tables in our database:

- create table teacher(id_teacher integer, name varchar(100), profile varchar(100), qualification varchar(100), about varchar(500), expertise text[], email varchar(100), phone_no varchar(100));
- create table teacher_course(id_teacher integer, code_module varchar(45), code_presentation varchar(45), subject_name varchar(45));
- create table subject_name(code_module varchar(45), subject_name varchar(45));
- create table course_commencement_date(code_presentation varchar(45), commencement_date date);
- create table site_name(id_site integer, site_name varchar(45));
- create table login_info(login_id text, password text, login_role text);
- create table student_basic(id_student integer, name varchar(45));
- create table student_main_info(id_student integer, name varchar(45), gender varchar(45), region varchar(45));

3.2.2 Constraints

Primary Keys

- ALTER TABLE courses ADD CONSTRAINT PK_courses PRIMARY KEY (code_module,code_presentation);
- ALTER TABLE assessments ADD CONSTRAINT PK_assessments PRIMARY KEY (code_module,code_presentation,id_assessment);
- ALTER TABLE vle ADD CONSTRAINT PK_vle PRIMARY KEY (id_site, code_module,code_presentation);
- ALTER TABLE studentInfo ADD CONSTRAINT PK_studentInfo PRIMARY KEY (code_module,code_presentation,id_student);
- ALTER TABLE studentRegistration ADD CONSTRAINT PK_studentRegistration PRIMARY KEY (code_module,code_presentation,id_student);
- ALTER TABLE studentAssessment ADD CONSTRAINT PK_studentAssessment PRIMARY KEY (id_assessment,id_student);
- ALTER TABLE studentVle ADD CONSTRAINT PK_studentVle PRIMARY KEY (id_studentVle);
- ALTER TABLE teacher ADD CONSTRAINT PK_teacher PRIMARY KEY (id_teacher);
- ALTER TABLE teacher_course ADD CONSTRAINT PK_teacher_course PRIMARY KEY (id_teacher,code_module,code_presentation);
- ALTER TABLE subject_name ADD CONSTRAINT PK_subject_name PRIMARY KEY (code_module);
- ALTER TABLE course_commencement_date ADD CONSTRAINT PK_course_commencement_date PRIMARY KEY (code_presentation);
- ALTER TABLE site_name ADD CONSTRAINT PK_site_name PRIMARY KEY (id_site);
- ALTER TABLE login_info ADD CONSTRAINT PK_login_info PRIMARY KEY (login_id);
- ALTER TABLE student_basic ADD CONSTRAINT PK_student_basic PRIMARY KEY (id_student);
- ALTER TABLE student_main_info ADD CONSTRAINT PK_student_main_info PRIMARY KEY (id_student);

Foreign Keys

- ALTER TABLE studentinfo
  ADD CONSTRAINT FK_studentinfo1
  FOREIGN KEY (code_module, code_presentation)
  REFERENCES courses(code_module, code_presentation) MATCH SIMPLE
  ON UPDATE NO ACTION ON DELETE NO ACTION;
- ALTER TABLE studentinfo
  ADD CONSTRAINT FK_studentinfo2
  FOREIGN KEY (id_student)
  REFERENCES student_main_info(id_student) MATCH SIMPLE
  ON UPDATE NO ACTION ON DELETE NO ACTION;
- ALTER TABLE assessments
  ADD CONSTRAINT FK_assessments1
  FOREIGN KEY (code_module, code_presentation)
  REFERENCES courses(code_module, code_presentation) MATCH SIMPLE
  ON UPDATE NO ACTION ON DELETE NO ACTION;

- ALTER TABLE studentregistration
  ADD CONSTRAINT FK_studentregistration1
  FOREIGN KEY (code_module, code_presentation)
  REFERENCES courses(code_module, code_presentation) MATCH SIMPLE
  ON UPDATE NO ACTION ON DELETE NO ACTION;
- ALTER TABLE studentregistration
  ADD CONSTRAINT FK_studentregistration2
  FOREIGN KEY (id_student)
  REFERENCES student_main_info(id_student) MATCH SIMPLE
  ON UPDATE NO ACTION ON DELETE NO ACTION;
- ALTER TABLE vle
  ADD CONSTRAINT FK_vle1
  FOREIGN KEY (code_module, code_presentation)
  REFERENCES courses(code_module, code_presentation) MATCH SIMPLE
  ON UPDATE NO ACTION ON DELETE NO ACTION;
- ALTER TABLE studentvle
  ADD CONSTRAINT FK_studentvle1
  FOREIGN KEY (code_module, code_presentation)
  REFERENCES courses(code_module, code_presentation) MATCH SIMPLE
  ON UPDATE NO ACTION ON DELETE NO ACTION;
- ALTER TABLE studentvle
  ADD CONSTRAINT FK_studentvle2
  FOREIGN KEY (id_student)
  REFERENCES student_main_info(id_student) MATCH SIMPLE
  ON UPDATE NO ACTION ON DELETE NO ACTION;
- ALTER TABLE studentassessment
  ADD CONSTRAINT FK_studentassessment1
  FOREIGN KEY (id_student)
  REFERENCES student_main_info(id_student) MATCH SIMPLE
  ON UPDATE NO ACTION ON DELETE NO ACTION;
- ALTER TABLE student_name_info
  ADD CONSTRAINT FK_student_name_info1
  FOREIGN KEY (id_student)
  REFERENCES student_main_info(id_student) MATCH SIMPLE
  ON UPDATE NO ACTION ON DELETE NO ACTION;
- ALTER TABLE teacher_course
  ADD CONSTRAINT FK_teacher_course1
  FOREIGN KEY (code_module, code_presentation)
  REFERENCES courses(code_module, code_presentation) MATCH SIMPLE
  ON UPDATE NO ACTION ON DELETE NO ACTION;
- ALTER TABLE teacher_course
  ADD CONSTRAINT FK_teacher_course2
  FOREIGN KEY (id_teacher)
  REFERENCES teacher(id_teacher) MATCH SIMPLE
  ON UPDATE NO ACTION ON DELETE NO ACTION;

Checks

- ALTER TABLE studentinfo
  ADD CONSTRAINT CHK_studentinfo1
  CHECK (gender='M' OR gender='F' OR gender='T');
- ALTER TABLE studentinfo
  ADD CONSTRAINT CHK_studentinfo2
  CHECK (disability='Y' OR disability='N');

- ALTER TABLE studentinfo
  ADD CONSTRAINT CHK_studentinfo3
  CHECK (age_band='55 $\leq$' OR age_band='0 – 35' OR age_band='35 – 55');
- ALTER TABLE studentinfo
  ADD CONSTRAINT CHK_studentinfo4
  CHECK (highest_education IN ('Lower Than A Level','A Level or Equivalent','HE Qualification','Post Graduate Qualification','No Formal quals'));
- ALTER TABLE studentinfo
  ADD CONSTRAINT CHK_studentinfo5
  CHECK (imd_band IN ('30-40%','40-50%', '60-70%','50-60%','0-10%','20-30%','10-20','80-90%','70-80%','90-100%'));
- ALTER TABLE studentinfo
  ADD CONSTRAINT CHK_studentinfo6
  CHECK (final_result IN ('Pass','Distinction','Withdrawn','Fail','Awaiting'));
- ALTER TABLE student_basic_info
  ADD CONSTRAINT CHK_student_basic_info
  CHECK (gender='M' OR gender='F' OR gender='T');
- ALTER TABLE studentassessment
  ADD CONSTRAINT CHK_studentassessment1
  CHECK (score$\geq$ 0 and score$\leq$ 100);
- ALTER TABLE studentassessment
  ADD CONSTRAINT CHK_studentassessment2
  CHECK (is_banked=0 or is_banked=1);
- ALTER TABLE assessments
  ADD CONSTRAINT CHK_assessments1
  CHECK (weight$\geq$ 0 and weight$\leq$ 100);
- ALTER TABLE assessments
  ADD CONSTRAINT CHK_assessments2
  CHECK (assessment_type='TMA' OR assessment_type='CMA' OR assessment_type='Exam');

3.2.3 Indexes

We built indexes on the following tables/attributes:

- CREATE INDEX idx_studentinfo ON studentinfo (code_module, code_presentation, id_student);
- CREATE INDEX idx_courses ON courses (code_module, code_presentation);
- CREATE INDEX idx_assessments ON assessments (code_module, code_presentation, id_assessment);
- CREATE INDEX idx_studentassessment ON studentassessment (id_assessment, id_student);
- CREATE INDEX idx_studentregistration ON studentregistration (code_module, code_presentation, id_student);
- CREATE INDEX idx_vle ON vle (id_site, code_module, code_presentation);
- CREATE INDEX idx_studentvle ON studentvle (id_studentvle);
- CREATE INDEX idx_student_main_info ON student_main_info (id_student);
- CREATE INDEX idx_student_name_info ON student_name_info (id_student);
- CREATE INDEX idx_course_commencement_date ON course_commencement_date (code_presentation);
- CREATE INDEX idx_teacher ON teacher (id_teacher);
- CREATE INDEX idx_teacher_course ON teacher_course (id_teacher, code_module, code_presentation);

3.2.4 Sequences

We built following sequences and used to generate next value for the ID attribute of tables :

- CREATE SEQUENCE seq_student_main_info INCREMENT BY 1 START WITH 2716796 NO MAXVALUE;
  ALTER TABLE student_main_info ALTER COLUMN id_student set default nextval('seq_student_main_info');
- CREATE SEQUENCE seq_assessments INCREMENT BY 1 START WITH 40089 NO MAXVALUE;
  ALTER TABLE assessments ALTER COLUMN id_assessment set default nextval ('seq_assessments');
- CREATE SEQUENCE seq_vle INCREMENT BY 1 START WITH 1077906 NO MAXVALUE;
  ALTER TABLE vle ALTER COLUMN id_site set default nextval ('seq_vle');
- CREATE SEQUENCE seq_assessments INCREMENT BY 1 START WITH 40089 NO MAXVALUE;
  ALTER TABLE assessments ALTER COLUMN id_assessment set default nextval ('seq_assessments');

3.2.5 Views

Following Materialized views have been created in database for query speed up:

- **Name: teacher_subject**
  Description: It contain information about all the subject taught by the teacher.
  Attributes: id_teacher, code_module, code_presentation, subject_name, teacher_name
  No. of tuples: 22
  Run Time (ms): 0.934 ms
- **Name: tr_sub_date**
  Description: It contain information about all the subject , course commencement date and course taught by the teacher.
  Attributes:id_teacher, code_module, code_presentation, subject_name , teacher_name, startdate
  No. of tuples: 22
  Run Time (ms): 0.620 ms
- **Name: tr_sub_dat_stid**
  Description: It contain information about all the student id, subject , course commencement date and course taught by the teacher
  Attributes:id_teacher, code_module, code_presentation, subject_name , teacher_name , startdate , id_student, final_result
  No. of tuples: 32593
  Run Time (ms): 58.416 ms
- **Name: tr_sub_dat_stid_stnm**
  Description: It contain information about all the student name, student id, subject , course commencement date and course taught by the teacher
  Attributes:id_teacher, code_module, code_presentation, subject_name , teacher_name , startdate , id_student, final_result, student_name
  No. of tuples: 32593
  Run Time (ms): 43.848 ms
- **Name: tr_sub_dat_stid_stnm_amt**
  Description: It contain information about all the student name, student id, subject , course commencement date , course taught by the teacher and assessment of the course
  Attributes:id_teacher, code_module, code_presentation, subject_name , teacher_name , startdate , id_student, final_result, student_name, id_assessment, assessment_type, assessmentdate, weight
  No. of tuples:323925
  Run Time (ms): 627.657 ms

- **Name: tr_sub_dat_stid_stnm_amt_stamt**
  Description: It contain information about all the student name, student id, subject , course commencement date , course taught by the teacher ,assessment of the course and performance of students in this assessments.
  Attributes:id_teacher, code_module, code_presentation, subject_name , teacher_name , startdate , id_student, final_result, student_name , id_assessment, assessment_type, assessmentdate, weight, asmtsubmissiondate, is_banked, score
  No. of tuples: 323925
  Run Time (ms): 624.904 ms

- **Name: not_happen_esment**
  Description: It contain list of the assessments that do not happen or no students participated in them.
  Attributes:code_module, code_presentation, id_assessment, assessment_type, date, weight
  No. of tuples: 18
  Run Time (ms): 1.111 ms

- **Name: happen_asment**
  Description: It contain list of the assessments that happen and students have participated in them.
  Attributes:code_module, code_presentation, id_assessment, assessment_type, date, weight
  No. of tuples: 188
  Run Time (ms): 3.565 ms

- **Name: tr_sub_dat_stid_stnm_hapamt**
  Description: It contain information about all the student name, student id, subject , course commencement date , course taught by the teacher and happened assessment of the course in which students have particpated.
  Attributes:id_teacher, code_module, code_presentation, subject_name , teacher_name , startdate , id_student, final_result, student_name , id_assessment, assessment_type, assessmentdate, weight
  No. of tuples: 297604
  Run Time (ms): 544.974 ms

- **Name: tr_sub_dat_stid_stnm_hapamt_stamt**
  Description: It contain information about all the student name, student id, subject , course commencement date , course taught by the teacher and happened assessment of the course in which students have particpated and their score of these student assessments.
  Attributes:id_teacher, code_module, code_presentation, subject_name , teacher_name , startdate , id_student, final_result, student_name , id_assessment, assessment_type, assessmentdate, weight, asmtsubmissiondate, is_banked, score
  No. of tuples: 297604
  Run Time (ms): 685.399 ms

- **Name: stud_wgt_hapass_score**
  Description: It contain score of the assessments of the students.
  Attributes:code_module, code_presentation, id_student, student_name , id_assessment, assessment_type, weight, score, aswgtscore
  No. of tuples: 297604
  Run Time (ms): 526.699 ms

- **Name: stud_asthaptype_agsco**
  Description: It contain aggregate score of the assessments of the students.
  Attributes:code_module, code_presentation, id_student, student_name, assessment_type, agrweight, agrscore
  No. of tuples: 64949
  Run Time (ms): 115.840 ms

- **Name: stud_ag_exam**
  Description: It contain aggregate score of the assessment type (- exam) of the students.
  Attributes:code_module, code_presentation, id_student, student_name , assessment_type, agrweight, agrscore

No. of tuples: 10706
Run Time (ms): 30.410 ms

- **Name: stud_ag_tma**
  Description: It contain aggregate score of the assessment type (- tma) of the students.
  Attributes:code_module, code_presentation, id_student, student_name, assessment_type, agrweight, agrscore
  No. of tuples: 32593
  Run Time (ms): 72.289 ms

- **Name: stud_ag_cma**
  Description: It contain aggregate score of the assessment type (- cma) of the students.
  Attributes:code_module, code_presentation, id_student, student_name, assessment_type, agrweight, agrscore
  No. of tuples: 21650
  Run Time (ms): 47.677 ms

- **Name: stud_ag_cmatma**
  Description: It contain aggregate score of the both assessment type (- tma and cma) of the students.
  Attributes:code_module, code_presentation, id_student, student_name , assessment_type, agrweight, agrscore
  No. of tuples: 32593
  Run Time (ms): 73.030 ms

- **Name: stud_ag_allscore**
  Description: It contain aggregate score of all the assessment type (- tma,cma and exam) of the students.
  Attributes:code_module, code_presentation, id_student, student_name, agrweight, agrscore, cnt
  No. of tuples: 32593
  Run Time (ms): 70.213 ms

- **Name: stud_ag_finalscore**
  Description: It contain final score of the students.
  Attributes:code_module, code_presentation, id_student, student_name, agrweight, agrscore, cnt, finalscore
  No. of tuples:32593
  Run Time (ms): 89.212 ms

- **Name: stud_fs_result**
  Description: It conatain final result of the students of all the courses.
  Attributes:code_module, code_presentation, id_student, student_name, agrweight, agrscore, cnt, finalscore, final_result
  No. of tuples: 32593
  Run Time (ms): 89.921 ms

- **Name: course_final_result**
  Description: It contain information about final result i.e. how many students are passed, failed or get distinction. lt of the co
  esurmat **Attributes:**code_module, code_presentation, final_result, total_student
  No. of tuples: 88
  Run Time (ms): 38.966 ms

- **Name: course_total_student**
  Description: It contain information about the strength of the course.

  Attributes:code_module, code_presentation, total_student
  No. of tuples: 22
  Run Time (ms): 0.413 ms

- **Name: get_stud_act_day_visit**
  Description: It contain inforamtion about students how many times they visited particular vle material of particular course on particular day.

Attributes:code_module, code_presentation, id_student, id_site, date, count
No. of tuples:8459320
Run Time (ms): 7880.859 ms

- **Name: get_stud_day_visit**
  Description: It contain inforamtion about students how many times they visited vle materials of particular course on particular day.
  Attributes:code_module, code_presentation, id_student, date, count
  No. of tuples: 1808119
  Run Time (ms): 1184.651 ms

- **Name: get_stud_visit**
  Description: It contain inforamtion about students how many times they visited particular vle material of particular coursse.
  Attributes:code_module, code_presentation, id_student, count
  No. of tuples: 29228
  Run Time (ms): 62.597 ms

- **Name: get_visit**
  Description: It contain inforamtion about total visits on the vle material of particular course.
  Attributes:code_module, code_presentation, count
  No. of tuples: 22
  Run Time (ms):0.579 ms

- **Name: vle_mat_cont**
  Description: It contain information about the count and different type of vle material of the particular course.
  Attributes:code_module, code_presentation, activity_type, count
  No. of tuples: 232
  Run Time (ms):3.413 ms

Above materialized views can be refreshed by following script:

```
REFRESH MATERIALIZED VIEW teacher_subject ;
REFRESH MATERIALIZED VIEW tr_sub_date ;
REFRESH MATERIALIZED VIEW tr_sub_dat_stid ;
REFRESH MATERIALIZED VIEW tr_sub_dat_stid_stnm ;
REFRESH MATERIALIZED VIEW tr_sub_dat_stid_stnm_amt ;
REFRESH MATERIALIZED VIEW tr_sub_dat_stid_stnm_amt_stamt ;
REFRESH MATERIALIZED VIEW not_happen_esment ;
REFRESH MATERIALIZED VIEW happen_asment ;
REFRESH MATERIALIZED VIEW tr_sub_dat_stid_stnm_hapamt ;
REFRESH MATERIALIZED VIEW tr_sub_dat_stid_stnm_hapamt_stamt ;
REFRESH MATERIALIZED VIEW stud_wgt_hapass_score ;
REFRESH MATERIALIZED VIEW stud_asthaptype_agsco ;
REFRESH MATERIALIZED VIEW stud_ag_exam ;
REFRESH MATERIALIZED VIEW stud_ag_tma ;
REFRESH MATERIALIZED VIEW stud_ag_cma ;
REFRESH MATERIALIZED VIEW stud_ag_cmatma ;
REFRESH MATERIALIZED VIEW stud_ag_allscore ;
REFRESH MATERIALIZED VIEW stud_ag_finalscore ;
REFRESH MATERIALIZED VIEW stud_fs_result ;
REFRESH MATERIALIZED VIEW course_final_result ;
REFRESH MATERIALIZED VIEW course_total_student ;
REFRESH MATERIALIZED VIEW get_stud_act_day_visit ;
REFRESH MATERIALIZED VIEW get_stud_day_visit ;
REFRESH MATERIALIZED VIEW get_stud_visit ;
REFRESH MATERIALIZED VIEW get_visit ;
REFRESH MATERIALIZED VIEW vle_mat_cont;
```

3.2.6 Triggers

The following Triggers have been created in database to perform specific task:

- To set final result withdrawn in studentinfo after student get unregistered:

```
CREATE OR REPLACE FUNCTION aft_update_unregister()
RETURNS trigger AS
$$
BEGIN
         IF NEW.date_unregistration is not null THEN
        update studentinfo set final_result='Withdrawn' where code_module=OLD.code_module and
code_presentation=OLD.code_presentation;
        END IF;
RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';
CREATE TRIGGER updt_final_result
        AFTER UPDATE ON studentregistration
        FOR EACH ROW
                EXECUTE PROCEDURE aft_update_unregister();
```

- To check that student cannot register after 180 days of module start time:

```
CREATE FUNCTION studentregistration_check() RETURNS
trigger AS $$
BEGIN
– Check that studentregistration info is correct
        IF (NEW.date_registration IS not NULL) and (NEW.date_registration ¿ 180) THEN
                RAISE EXCEPTION 'Student cannot register after 180 days of module start time';
        END IF;
RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';


CREATE TRIGGER studentregistration_check_tigger BEFORE INSERT OR UPDATE ON studen-
tregistration
        FOR EACH ROW EXECUTE PROCEDURE
                studentregistration_check();
```

- To check that cut off date of exam or assessments must be less than module presentation length

```
CREATE FUNCTION assessments_check() RETURNS trigger AS
$$ BEGIN
– Check that assessments info is correct
        IF (NEW.date ¿ (select length from courses where code_module=OLD.code_module and code_-
presentation=OLD.code_presentation) ) THEN
                RAISE EXCEPTION 'Assessment date must be within course duration';
        END IF;
RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';

CREATE TRIGGER assessments_check_tigger BEFORE INSERT OR UPDATE ON assessments
        FOR EACH ROW EXECUTE PROCEDURE                  assessments_check();
```

- To initialize no_of_prev_attempt=0, final_result='Awaiting' after new insert into studentinfo table:

```
CREATE OR REPLACE FUNCTION aft_insert_studentinfo()
RETURNS trigger AS
$$
BEGIN
    IF NEW.final_result is null THEN
update studentinfo set final_result='Awaiting' where code_module=OLD.code_module and code_pre-
sentation=OLD.code_presentation;
    ELSIF NEW.no_of_prev_attempt is null THEN
        update studentinfo set no_of_prev_attempt=0 where code_module=OLD.code_module and
code_presentation=OLD.code_presentation;
    END IF;
RETURN NEW;
END;
$$
LANGUAGE 'plpgsql';

CREATE TRIGGER updt_studentinfo
AFTER INSERT ON studentregistration
    FOR EACH ROW
        EXECUTE PROCEDURE aft_insert_studentinfo();
```

- Trigger to check login info:

```
CREATE FUNCTION login_check() RETURNS trigger AS $$
BEGIN
– Check that login_id and pwd are given
    IF NEW.login_id IS NULL THEN
        RAISE EXCEPTION 'login_id cannot be null or blank';
    END IF;
    IF NEW.password IS NULL THEN
        RAISE EXCEPTION 'password cannot be null or blank';
    END IF;
    IF LENGTH(NEW.password)¡ 5 THEN
        RAISE EXCEPTION 'password cannot be less than 5 characters';
    END IF;
    IF NEW.login_role not in('Admin','Student','Teacher') THEN
        RAISE EXCEPTION 'login_role must be Admin or Student or Teacher';
    END IF;
RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';

CREATE TRIGGER login_check_tigger BEFORE INSERT OR UPDATE
ON login_info
    FOR EACH ROW EXECUTE PROCEDURE login_check();
```

3.2.7 Procedures/ UDF(User Defined Functions)

Function	Details
getTeacherSubjects	**Input:** teacherid **Output:** Table of (subjects_id,subject_name)
getSubjectTeachers	**Input:** Codemodule **Output:** Table of (teacher_id,teacher_name)
getMaxEnroll	**Input:** code_presentation **Output:** list of (code_module,subject_name,subject_-teacher, max_enroll_num) **Description :** Gives course which has maximum enrollment for particular code_presenation
getMinEnroll	**Input :** code_presentation **Output :** list of (code_module,subject_name,subject_-teacher, max_enroll_num) **Description :** Gives course which has minimum enrollment for particular code_presentation
getResultWiseStudentCount	**Input :** code_module,code_presentation **Output :** table of (final_result,total_student) **Description :** Gives resultwise distribution for particular course
getDurationOfCourseModule	**Input :** code_module,code_presentation **Output :** Duration of coursemodule
getMaxDurationCourseModule	**Input :** code_presentation **Output :** table of (code_module,subjectname,length) **Description :** Gives courses with subject name which has maximum length for any code_presentation
getNoOfAssessmentInCourse	**Input :** code_module,code_presentation **Output :** list of(assessment_type,noOfAssessment) **Description :** gives number of each type of assessment in the course
getWeightageInfoOfAssessmentInCourse	**Input :** code_module,code_presentation, assessment_type **Output :** table of (assessmentID,weightage) **Description** gives all assessmentid and their weightage for particular course and assessmenttype
getWeightageInfoOfAssessmentInCourse	**Input :** code_module,code_presentation **Output :** table of (assessID,assessment_type,weightage) **Description** return all the assessmentid with their types and weightages.
getCourseVLEType	**Input :** code_module,code_presentation **Output :** table of (id_site,activity_type) **Description :** gives ids of vle material and their activity types for a particular coours
getNoOfCourseVLETypeWise	**Input :** code_module,code_presentation **Output :** Table of (activity_type, count) **Description :** return all the activity type of vle with their count for the particular course.
getStudentGenderCount	**Input :** code_module,code_presentation **Output :** table of (gender, count) **Description :** gives count of male and female students in the particular course.
getNoOfStudentRegionWise	**Input :** code_module,code_presentation **Output :** table of (region, count) **Description :** Return count of the student regionwise in the course.
getNoOfStudentRegionWise	**Input :** code_module,code_presentation,region

	Output : Number of student which belong to given region for given course.
getNoOfStudentHtEducationWise	**Input :** code_module,code_presentation **Output :** table of (highest_education, count) **Description :** Return number of students grouped according to their highest education .
getNoOfStudentHtEducationWise	**Input:** code_module,code_presentation,highest_education **Output :** Return count of students with given highest education for given course.
getNoOfStudentIMDBandWise	**Input :** code_module,code_presentation **Output :** table of (imdband ,count) **Description :** Return count of student for each type of imd band for particular course.
getNoOfStudentIMDBandWise	**Input :** code_module,code_presentation,imd_band **Output :** Gives count of student who belong to given imd-band for given course.
getNoOfStudentAgeBandWise	**Input :** code_module,code_presentation **Output :** table of(age_band, count) **Description :** Return count of student for each type of age band for particular course.
getNoOfStudentAgeBandWise getNoOfStudentDisabled	**Input :** code_module,code_presentation,age_band **Output :** Gives count of student who belong to given age band for given course. **Input :** code_module,code_presentation **Output :** Count of student who are disabled and enrolled in given course.
getNoOfStudentRegisteredBeforeCourseStart	**Input :** code_module,code_presentation **Output :** Gives count of student who have registered for given course before date of start of course.
getNoOfStudentRegisteredAfterCourseStart	**Input :** code_module,code_presentation **Output :** Gives count of student who have registered for given course after date of start of course.
getNoOfStudentWithdrawnCourse	**Input :** code_module,code_presentation **Output :** Gives count of students who have withdrawn from the course.
get_course_strength	**Input:** code_module,code_presentation **Output :** Gives total student in that course.
get_course_final_result	**Input :** code_module,code_presentation **Output :** table of (finalresult,count) **Description :** gives statistics about the pass,distinction,withdrawn or failed students in the given course.
get_course_student	**Input :** code_module,code_presentation **Output :** table of (studentid,studentname,result) **Description :** Return the information about student in the course in ordering of their result.
getPercentageOfStudentResultWise	**Input :** Studentinfo table **Output :** Return percentwise result distribution of all the student in the given dataset.
getPercentageOfStudentResultWise	**Input :** code_module,code_presentation **Output :** table of (finalresult,percentage) **Description :** gives resultwise distribution of student in given course.
getTeacherWhoIsNotTeachingAnyCourse	**Input :** code_presentation **Output :** table of (id_teacher, name) **Description:** return id and name of teacher who are not teaching any subject in given course presentation.

getTeacherAndCourseCount	**Input :** code_presentation **Output :** table of (id_teacher, name,coursecount) **Description :** gives id ,name of teachers and their subject that they are teaching in that code presentation.
fail_stud_codewise	**Input :** code_module,code_presentation,cnt **Output :** table of (id_student,student_name,finalscore) **Description :** Return list of failed student with their final score for given course
fail_stud_cnt_codewise	**Input :** code_module,code_presentation **Output :** Count of student who have failed in given course.
top_stud_codwise	**Input :** code_module,code_presentation,cnt **Output :** table of (id_student integer,student_name,finalscore) **Description :** gives list of id and names of top (given cnt) student for given course.
great_stud_codwise	**Input :** code_module,code_presentation,score **Output :** table of (id_student ,student_name,finalscore) **Description :** gives information about student who have get finalscore more than or equal to given value for given course.
great_stud_cnt_codwise	**Input :** code_module,code_presentation,score **Output :** Give count of student who have scored more than or equal to give score for given course.
lower_stud_codwise	**Input :** code_module,code_presentation,score **Output :** table of (id_student ,student_name,finalscore) **Description :** gives information about student who have scored lower than given score for given course.
lower_stud_cnt_codwise	**Input :** code_module,code_presentation,score **Output :** Give count of student who have scored more than or equal to give score for given course.
range_stud_codwise	**Input :** code_module,code_presentation,score1,score2 **Output :** table of (id_student ,student_name,finalscore) **Description :** gives information about students whose final score are between given two score for given course.
range_stud_cnt_codwise	**Input :** code_module,code_presentation,score1,score2 **Output :** gives count of students who have final score between two scores given for given course.
get_course_desval	**Input :** code_module,code_presentation,desired value **Output :** gives average,minimum or average of final score according to the desired value for given course.
get_course_studperfor	**Input :** code_module,code_presentation **Output :** return summary of student performance (average,minimum and maximum) based on their final score for given course.
get_course_amt_wgt_type	**Input :** code_module,code_presentation **Output :** table of (assessmenttype,agregate weightage) **Description :** return all the possible assessment type and their aggregate weightage.
get_top_course_agr_asstype	**Input :** code_module,code_presentation,assessment_type,cnt **Output :** table of (id_student,student_name,agrscore) **Description :** gives information about top (given cnt) student for given assessment type for given course.
get_low_course_agr_asstype	**Input :** code_module,code_presentation,assessment_type,cnt **Output :** table of (id_student,student_name,agrscore)

	Description : gives information about (given cnt) student who have scored low for given assessment type for given course.
get_course_exam_desval	**Input :** code_module,code_presentation,desired value **Output :** gives average,minimum or average of aggregate score according to the desired value for given assessment type (exam) given course.
get_course_tma_desval	**Input :** code_module,code_presentation,desired value **Output :** gives average,minimum or average of aggregate score according to the desired value for given assessment type (TMA) given course.
get_course_cma_desval	**Input :** code_module,code_presentation,desired value **Output :** gives average,minimum or average of aggregate score according to the desired value for given assessment type (CMA) given course.
get_course_cmatma_desval	**Input :** code_module,code_presentation,desired value **Output :** gives average,minimum or average of aggregate score for both cma and tma combined according to the desired value for given course.
get_course_exam$_studperfor$	**Input :** code_module,code_presentation **Output :** return summary (average,minimum,maximum) of assessment type- exam based on aggregate score of students for given course.
get_course_tma_studperfor	**Input :** code_module,code_presentation **Output :** return summary (average,minimum,maximum) of assessment type - TMA based on aggregate score of students for given course.
get_course_cma_studperfor	**Input :** code_module,code_presentation **Output :** return summary (average,minimum,maximum) of assessment type - CMA based on aggregate score of students for given course.
get_course_cmatma_studperfor	**Input :** code_module,code_presentation **Output :** return summary (average,minimum,maximum) of both assessment type (CMA and TMA) based on aggregate score of students for given course.
comp_course_gender	**Input :** code_module1,code_presentation1,gender1,code_module2,code_presentation2,gender2 **Output :** return string showing which course has higher value for given gender **Description :** Return course who has greater count for given arguments of gender type based on arguments of two courses given.
comp_course_region	**Input :** code_module1,code_presentation1,region1,code_module2,code_presentation2,region2 **Output :** return string showing which course has higher value for given region **Description :** Return course who has greater count for given arguments of regions type based on arguments of two courses given.
comp_course_highest_education	**Input :** code_module1,code_presentation1,highest_education1,code_module2,code_presentation2,high_education2 **Output :** return string showing which course has higher value for given arguments of highest education **Description :** Return course who has greater count for given arguments of highest education type based on arguments of two courses given.

comp_course_imd_band	**Input :** code_module1,code_presentation1,imd_band1,code_module2,code_presentation2,imd_band2 **Output :** return string showing which course has higher value for given arguments of imd band **Description :** Return course who has greater count for given arguments of imd band type based on arguments of two courses given.
comp_course_age_band	**Input :** code_module1,code_presentation1,age_band1,code_module2,code_presentation2,age_band2 **Output :** return string showing which course has higher value for given arguments of age band **Description :** Return course who has greater count for given arguments of age band type based on arguments of two courses given.
comp_course_disability	**Input :** code_module1,code_presentation1,disability1,code_module2,code_presentation2,disability2 **Output :** return string showing which course has higher value for given arguments of disability **Description :** Return course who has greater count for given arguments of disability(Y or N) based on arguments of two courses given.
assign_grade	**Input :** Materialized view stud_fs_result **Output :** Table stud_grade **Description :** used to make table stud_grade(has grade of student for their course) by the finalscore and finalresult of view stud_fs_result
get_course_amtlist	**Input :** code_module,code_presentation **Output :** table of (id_assessment,assessment_type,weight) **Description :** Return all the assessment id with their types and weightages for given course.
get_top_course_amt_stud	**Input :** code_module,code_presentation,id_assessment,cnt **Output :** table of (id_student,student_name,score) **Description :** Return top (given cnt) students (id and name) who have scored for given assessment id for given course.
get_low_course_amt_stud	**Input :** code_module,code_presentation,id_assessment,cnt **Output :** table of (id_student,student_name,score) **Description :** Return (given cnt) students (id and name) who have scored lowest for given assessment id for given course.
get_course_amt_desval	**Input :** code_module,code_presentation,id_assessment,desval **Output :** gives average,minimum or maximum of score of students for given assessment ids according to the desired value for given course.
get_course_amt_studperfor	**Input :** code_module,code_presentation,id_assessment **Output :** gives summary (average,minimum or maximum) score of students for given assessment of given course.
get_course_vle_cnt	**Input :** code_module,code_presentation **Output :** table of (activity_type,count) **Description :** Gives count of each type of vle material on the basis of activity type of vle.
stud_act_day_visit	**Input :** code_module,code_presentation,id_student,id_site,date

	Output : Count of given site of vle material visited by student with given student id on given day for given course.
stud_day_visit	**Input :** code_module,code_presentation,id_student,date **Output :** Total Count of visits on vle material of given course by student with given student id on given date
stud_visit	**Input :** code_module,code_presentation,id_student **Output :** Total count of visits on vle material of given course by student with given student id
coursevle_visit	**Input :** code_module,code_presentation **Output :** Total count of visits on vle material of given course

3.3 List of queries and run times

- **Query:** select * from get_subject_teachers('CCC');
  Run Time (ms): 1.054 ms
- **Query:** select * from getMaxEnroll('2013B');
  Run Time (ms): 0.783 ms
- **Query:** select * from getResultWiseStudentCount('CCC','2014B');
  Run Time (ms): 0.777 ms
- **Query:** select * from getDurationOfCourseModule('CCC','2014B');
  Run Time (ms): 0.791 ms
- **Query:** select * from getMaxDurationCourseModule('2014B');
  Run Time (ms): 4.031 ms
- **Query:** select * from getNoOfAssessmentInCourse('CCC','2014B');
  Run Time (ms): 0.841 ms
- **Query:** select * from getWeightageInfoOfAssessmentInCourse('CCC','2014B');
  Run Time (ms): 0.766 ms
- **Query:** select * from getCourseVLEType('CCC','2014B');
  Run Time (ms): 3.941 ms
- **Query:** select * from getNoOfCourseVLETypeWise('CCC','2014B');
  Run Time (ms): 2.396 ms
- **Query:** select * from getStudentGenderCount('CCC','2014B');
  Run Time (ms): 2.527 ms
- **Query:** select * from getNoOfStudentHtEducationWise('CCC','2014B');
  Run Time (ms): 2.643 ms
- **Query:** select * from getNoOfStudentAgeBandWise('CCC','2014B');
  Run Time (ms): 2.452 ms
- **Query:** select * from getNoOfStudentDisabled('CCC','2014B');
  Run Time (ms): 2.257 ms
- **Query:** select * from getNoOfStudentRegisteredAfterCourseStart('CCC','2014B');
  Run Time (ms): 2.479 ms
- **Query:** select * from getNoOfStudentWithdrawnCourse('CCC','2014B');
  Run Time (ms): 1.891 ms
- **Query:** select * from get_course_strength('CCC','2014B');
  Run Time (ms): 0.522 ms
- **Query:** select * from get_course_final_result('CCC','2014B');
  Run Time (ms): 1.041 ms

- **Query:** select * from get_course_student('CCC','2014B');
  Run Time (ms): 13.913 ms
- **Query:** select * from getPercentageOfStudentResultWise();
  Run Time (ms): 17.640 ms
- **Query:** select * from getPercentageOfStudentResultWise('CCC','2014B');
  Run Time (ms): 3.863 ms
- **Query:** select * from getTeachersWhoIsNotTeachingAnyCourse('2014B');
  Run Time (ms): 0.821 ms
- **Query:** select * from getTeacherAndCourseCount('2014B');
  Run Time (ms): 0.789 ms
- **Query:** select * from fail_stud_cnt_codewise('CCC','2014B');
  Run Time (ms): 9.946 ms
- **Query:** select * from fail_stud_cnt_codewise('CCC','2014B');
  Run Time (ms): 9.990 ms
- **Query:** select * from great_stud_cnt_codwise('CCC','2014B',10);
  Run Time (ms): 9.910 ms
- **Query:** select * from lower_stud_cnt_codwise('CCC','2014B',10);
  Run Time (ms): 9.102 ms
- **Query:** select * from get_course_exam_studperfor('CCC','2014B');
  Run Time (ms): 9.762 ms
- **Query:** select * from get_course_tma_studperfor('CCC','2014B');
  Run Time (ms): 16.744 ms
- **Query:**select * from get_course_cma_studperfor('CCC','2014B');
  Run Time (ms): 13.085 ms
- **Query:** select * from get_course_cmatma_studperfor('CCC','2014B');
  Run Time (ms): 18.560 ms
- **Query:** select * from get_course_vle_cnt('CCC','2014B');
  Run Time (ms): 0.580 ms
- **Query:** select * from stud_visit('AAA','2013J',11391);
  Run Time (ms): 0.490 ms
- **Query:** select * from coursevle_visit('AAA','2013J');
  Run Time (ms): 0.500 ms
- **Query:** select $\star fromlogin\_infowherelogin\_id =' romanholmes625743@gmail.com'andpassword =' 12345'andlogin\_role =' Student'$;
  Run Time (ms):$1.404ms$

4 ER Diagram

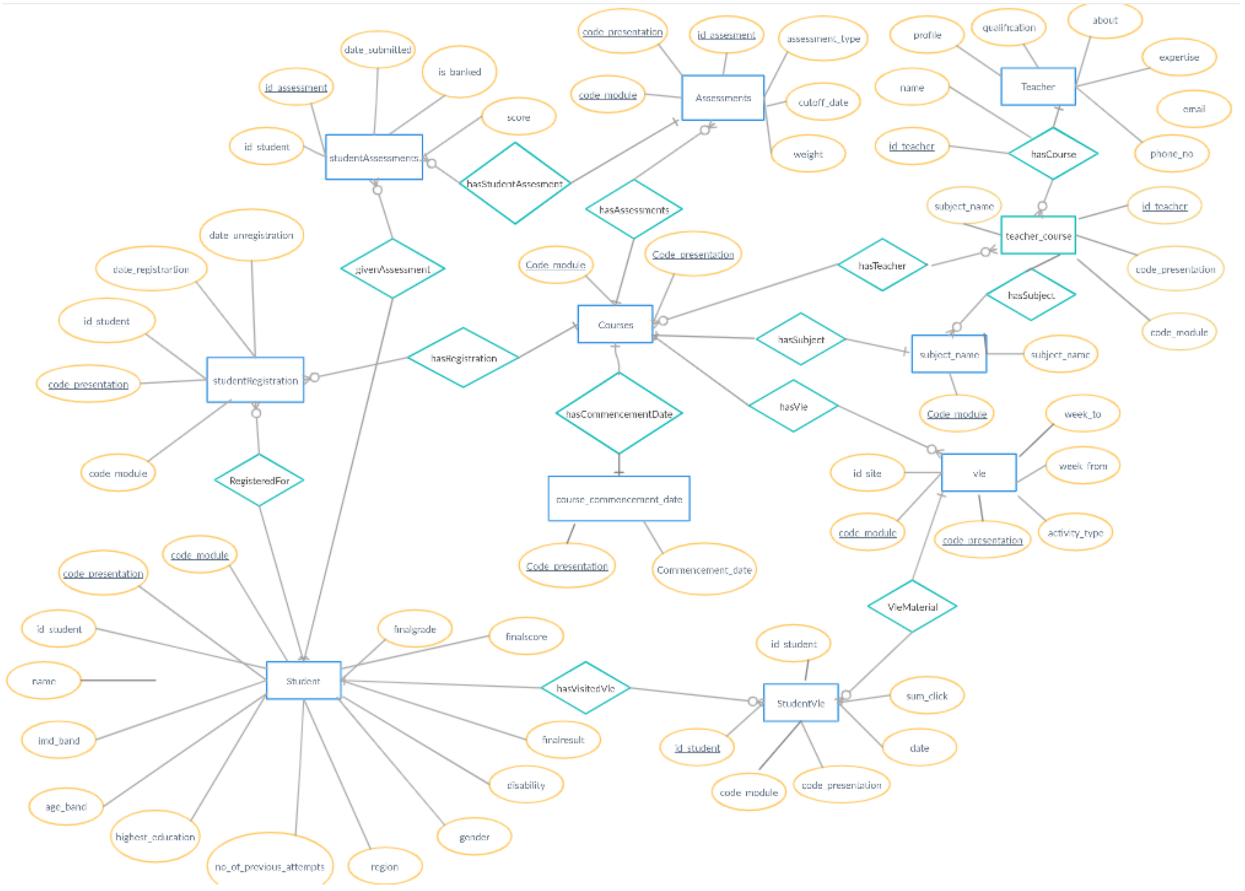

Figure 3: ER Diagram

5 Misc Points

- In this project, we have made use of the user defined functions.In this, we have used *if-else*,*for*, *array* and many aggregate functions.
- Also,We have used sequence for id generation and have implemented checks and triggers to satisfy required conditions.
- Tables like Subject_name,teacher,teacher_course,course_commencement_date,login_info , stud_grade etc are made to fulfill project requirements while the integrity of the project is maintained.

References

- Kuzilek J., Hlosta M., Zdrahal Z. Open University Learning Analytics dataset Sci. Data 4:170171 doi: 10.1038/sdata.2017.171 (2017).
- Papamitsiou, Z. Economides, A. A. Learning Analytics and Educational Data Mining in Practice: A Systematic Literature Review of Empirical Evidence. Educational Technology Society 17, 49–64 (2014).
- Kuzilek, J., Hlosta, M., Zdrahal, Z. figshare `https://doi.org/10.6084/m9.figshare.5081998.v1` (2017)
- `https://www.postgresql.org/docs/manuals/`