A Case Study of GITAM University Database

By Sohan Patchigolla

Abstract

In this case study on University Database, we have depicted the various relations in GITAM. In GITAM database, we start by taking a class called **University** which helps us to differentiate the data flow between the three campuses. Next, we have the various **Departments** on each campus ranging from CSE to Medical. Each department has its own **Faculty** members ***teaching*** various **Subjects**.
Now comes the most important entity in the entire University, **Students**. The students **Enroll** in various subjects taught by the respective faculty. Furthermore, the students are ***graded*** based on their performance in their respective **Examinations**.
We have also created **Student Log** relation to keep track of updates to the student database.
Finally, we have the relation **Accounts** between the student and the university. The students pay fee as per their course and scholarship.
We created this particular case study as a specific model for GITAM university but it can be scaled up for more universities with the same model design.

Part-I- Data Modeling Design

a) Data Requirements – Entities, Relationships, Attributes, Constraints.

a. Entities

i. University
ii. Department
iii. Faculty
iv. Subject
v. Teaches
vi. Student
vii. Studlog
viii. Exam
ix. Enrolled
x. Grades
xi. Accounts

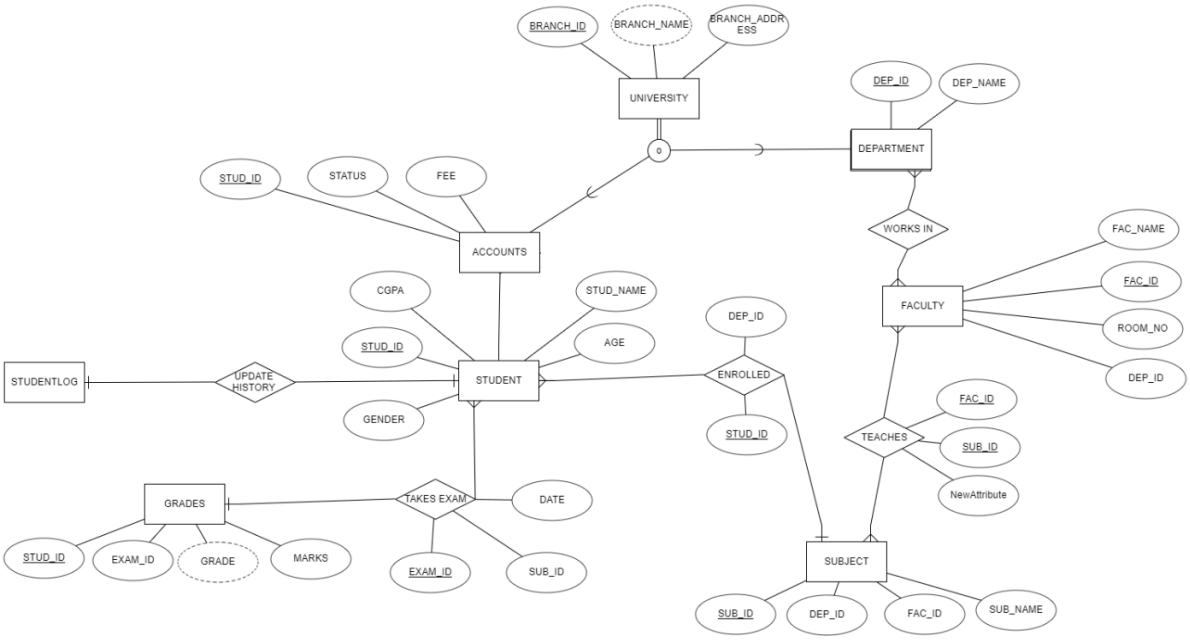
b. Relationships

i. Faculty work in departments
ii. Students enroll in subject
iii. Faculty teach student
iv. Students take exam and get grades
v. Students pay fees to the university

c. Attributes

I. UNIVERSITY(BRANCH_ID, BRANCH_NAME, BRANCH_ADDRESS)
II. DEPARTMENT(DEP_I, DEP_NAME)
III. FACULTY(FAC_I, FAC_NAME, ROOM_NO, DEP_ID)
IV. SUBJECT(SUB_ID, SUB_NAME, FAC_ID, DEP_ID)
V. TEACHES(FAC_ID, , SUB_ID)
VI. STUDENT(STUD_ID, STUD_NAME, , GENDER, CGPA, AGE)
VII. STUDLOG(STUD_ID, STUD_NAME, , GENDER, CGPA, AGE)
VIII. EXAM(EXAM_ID, SUB_ID, DATE)
IX. ENROLLED(STUDENT_ID VARCHAR, DEP_ID)
X. GRADES(STUD_ID, EXAM_ID, MARKS, GRADE)
XI. ACCOUNTS(STUD_ID, FEE, STATUS)

b) ER Diagram


BRANCH_ID
BRANCH_NAME
BRANCH_ADDRESS
UNIVERSITY
DEP_ID
DEP_NAME
DEPARTMENT
STUD_ID
STATUS
FEE
ACCOUNTS
WORKS IN
FAC_NAME
FAC_ID
ROOM_NO
DEP_ID
FACULTY
CGPA
STUD_NAME
STUD_ID
AGE
GENDER
STUDENT
STUDENTLOG
UPDATE HISTORY
DEP_ID
ENROLLED
STUD_ID
FAC_ID
TEACHES
SUB_ID
NewAttribute
GRADES
TAKES EXAM
DATE
STUD_ID
EXAM_ID
GRADE
MARKS
EXAM_ID
SUB_ID
SUBJECT
SUB_ID
DEP_ID
FAC_ID
SUB_NAME

Part-II-Relational Database Schema Development

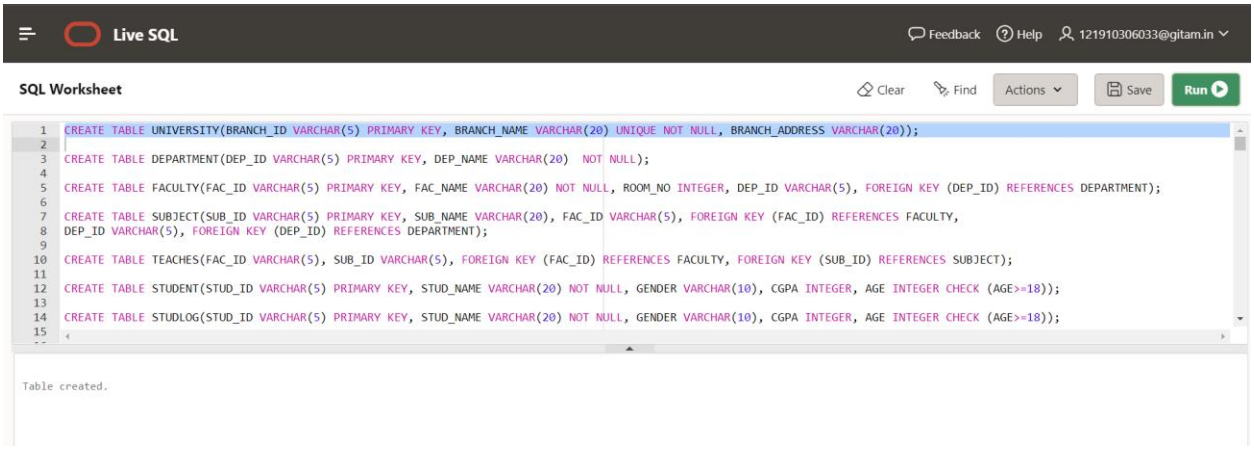
Conceptual Schema

1) UNIVERSITY(BRANCH_ID VARCHAR, BRANCH_NAME VARCHAR, BRANCH_ADDRESS VARCHAR)
2) DEPARTMENT(DEP_ID VARCHAR, DEP_NAME VARCHAR)
3) FACULTY(FAC_ID VARCHAR, FAC_NAME VARCHAR, ROOM_NO INTEGER, DEP_ID VARCHAR)
4) SUBJECT(SUB_ID VARCHAR, SUB_NAME VARCHAR, FAC_ID VARCHAR, DEP_ID VARCHAR)
5) TEACHES(FAC_ID, VARCHAR, SUB_ID VARCHAR)
6) STUDENT(STUD_ID VARCHAR, STUD_NAME, VARCHAR, GENDER VARCHAR, CGPA REAL, AGE INTEGER)
7) STUDLOG(STUD_ID VARCHAR, STUD_NAME, VARCHAR, GENDER VARCHAR, CGPA REAL, AGE INTEGER)
8) EXAM(EXAM_ID VARCHAR, SUB_ID VARCHAR, DATE DATE)
9) ENROLLED(STUDENT_ID VARCHAR, DEP_ID VARCHAR)
10) GRADES(STUD_ID VARCHAR, EXAM_ID VARCHAR, MARKS INTEGER, GRADE VARCHAR)
11) ACCOUNTS(STUD_ID VARCHAR, FEE INTEGER, STATUS VARCHAR)

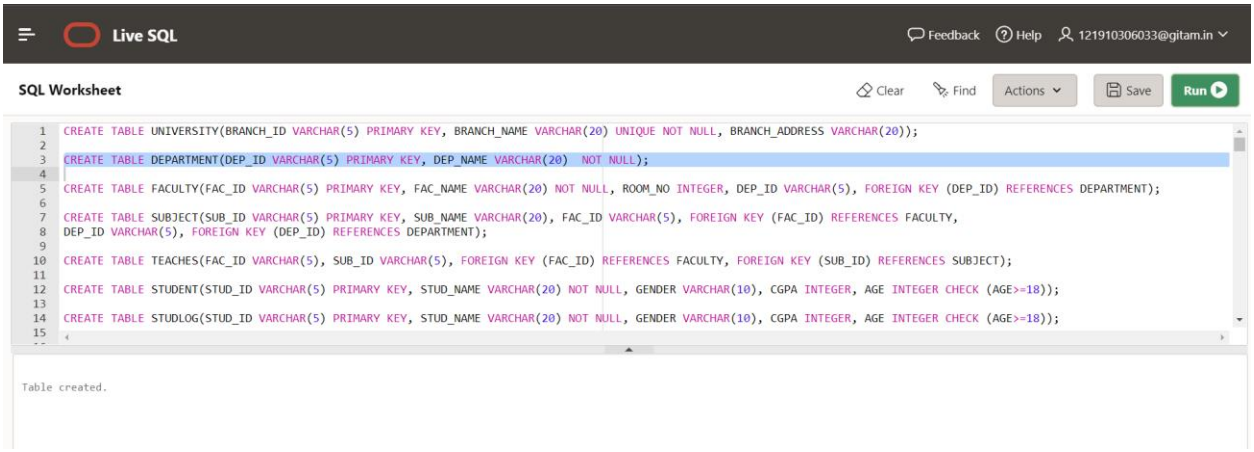
Part-III- Relational Database Implementation

- **Tables Creation**

CREATE TABLE UNIVERSITY(BRANCH_ID VARCHAR(5) PRIMARY KEY, BRANCH_NAME VARCHAR(20) UNIQUE NOT NULL, BRANCH_ADDRESS VARCHAR(20));

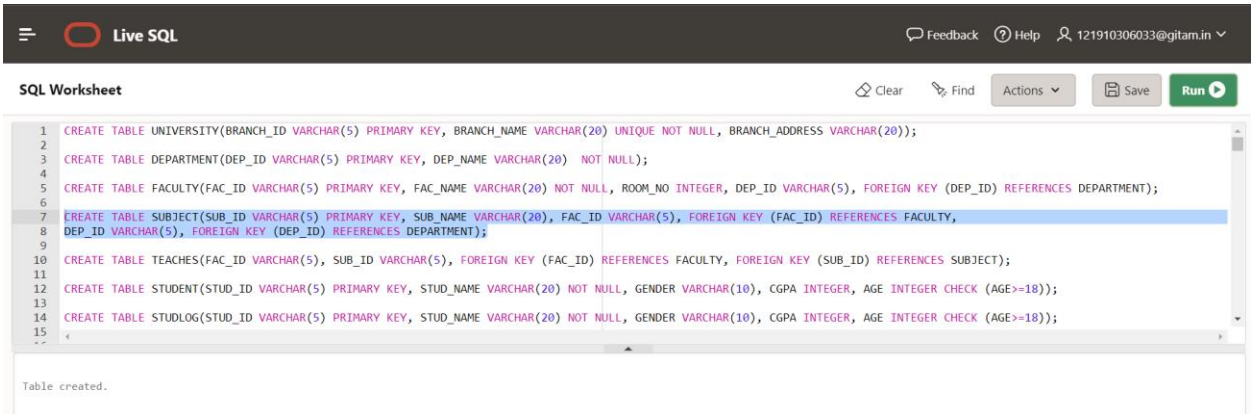


CREATE TABLE DEPARTMENT(DEP_ID VARCHAR(5) PRIMARY KEY, DEP_NAME VARCHAR(20) NOT NULL);

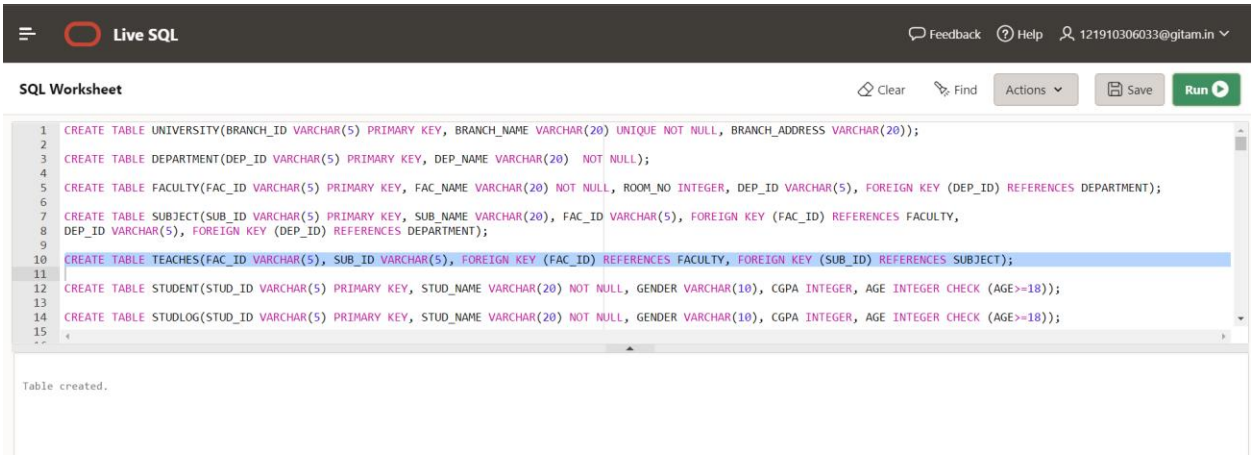


CREATE TABLE FACULTY(FAC_ID VARCHAR(5) PRIMARY KEY, FAC_NAME VARCHAR(20) NOT NULL, ROOM_NO INTEGER, DEP_ID VARCHAR(5), FOREIGN KEY (DEP_ID) REFERENCES DEPARTMENT);

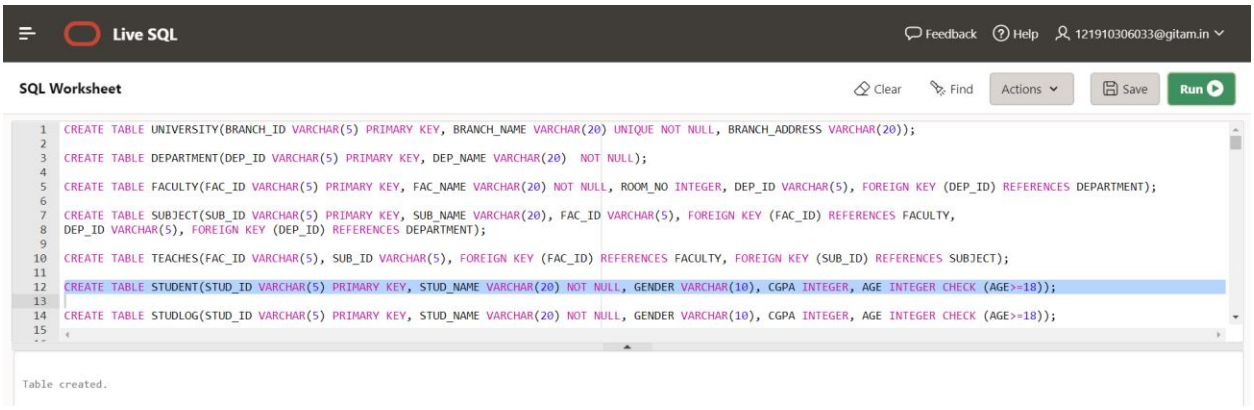
CREATE TABLE SUBJECT(SUB_ID VARCHAR(5) PRIMARY KEY, SUB_NAME VARCHAR(20), FAC_ID VARCHAR(5), FOREIGN KEY (FAC_ID) REFERENCES FACULTY,
DEP_ID VARCHAR(5), FOREIGN KEY (DEP_ID) REFERENCES DEPARTMENT);



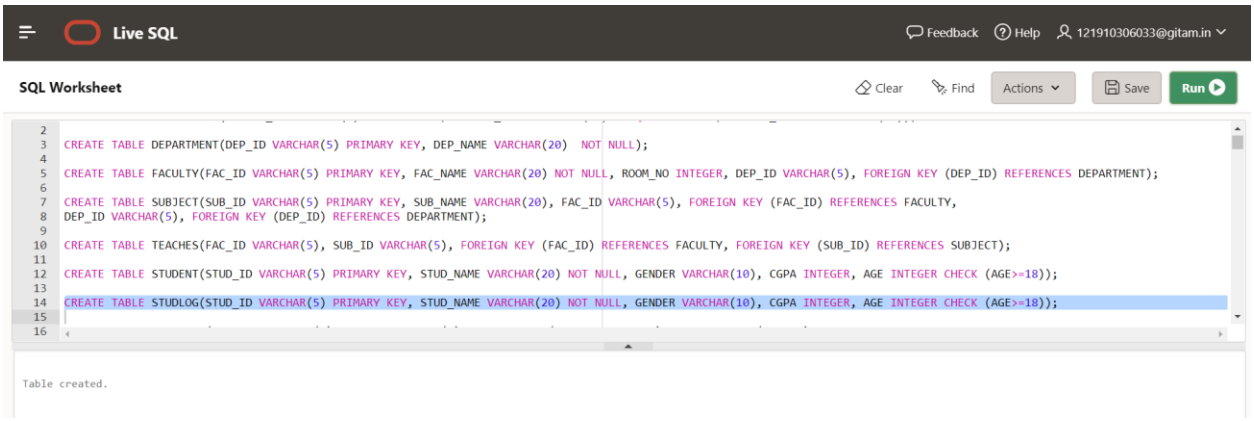
CREATE TABLE TEACHES(FAC_ID VARCHAR(5), SUB_ID VARCHAR(5), FOREIGN KEY (FAC_ID) REFERENCES FACULTY, FOREIGN KEY (SUB_ID) REFERENCES SUBJECT);



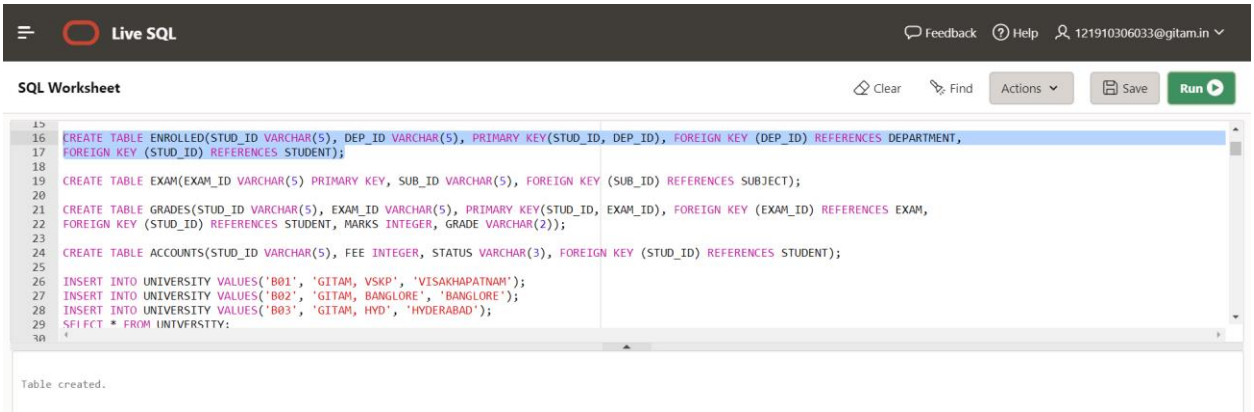
CREATE TABLE STUDENT(STUD_ID VARCHAR(5) PRIMARY KEY, STUD_NAME VARCHAR(20) NOT NULL, GENDER VARCHAR(10), CGPA INTEGER, AGE INTEGER CHECK (AGE>=18));



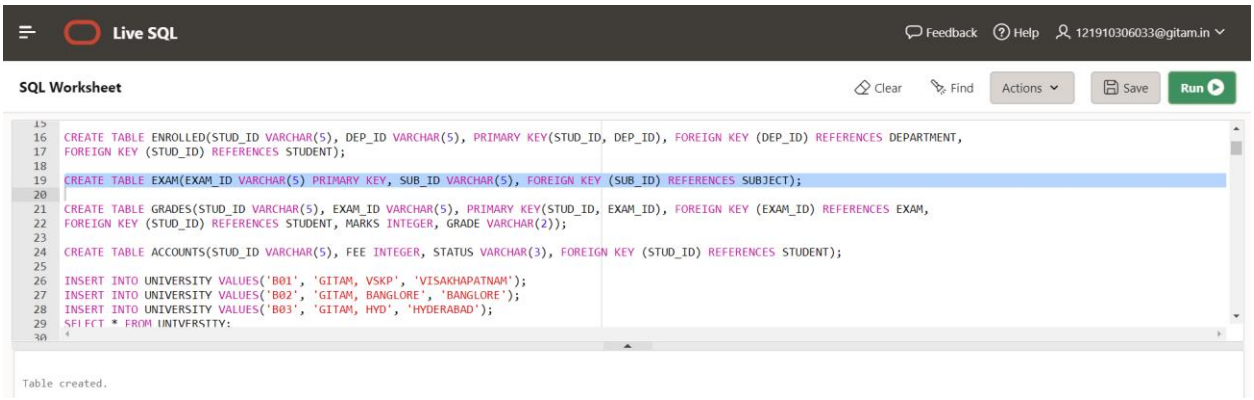
CREATE TABLE STUDLOG(STUD_ID VARCHAR(5) PRIMARY KEY, STUD_NAME VARCHAR(20) NOT NULL, GENDER VARCHAR(10), CGPA INTEGER, AGE INTEGER CHECK (AGE>=18));



CREATE TABLE ENROLLED(STUD_ID VARCHAR(5), DEP_ID VARCHAR(5), PRIMARY KEY(STUD_ID, DEP_ID), FOREIGN KEY (DEP_ID) REFERENCES DEPARTMENT,
FOREIGN KEY (STUD_ID) REFERENCES STUDENT);

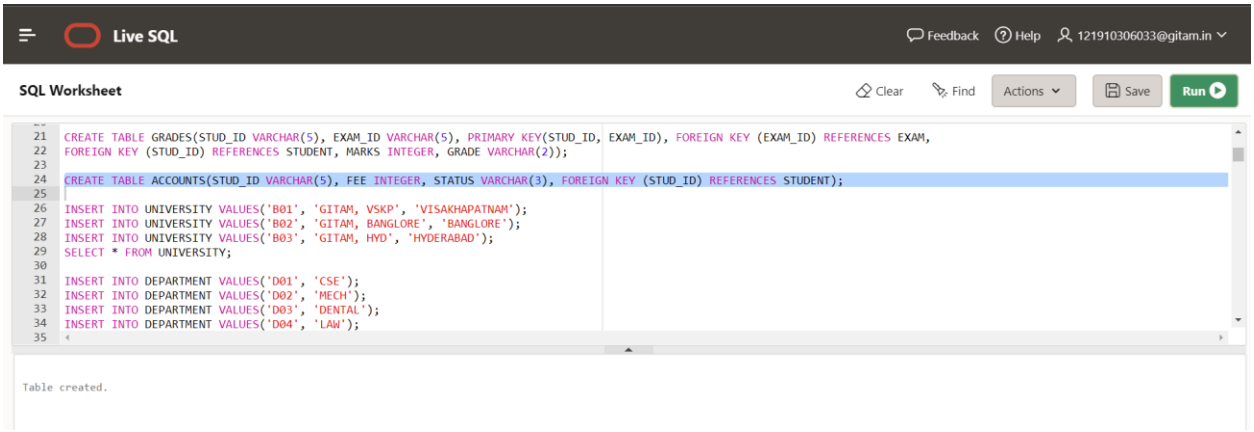


CREATE TABLE EXAM(EXAM_ID VARCHAR(5) PRIMARY KEY, SUB_ID VARCHAR(5), FOREIGN KEY (SUB_ID) REFERENCES SUBJECT);



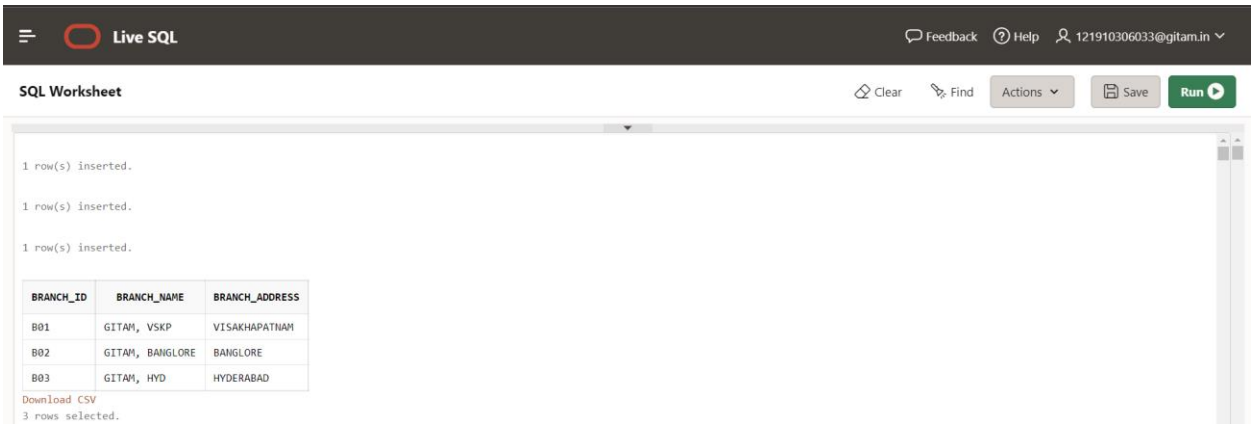
CREATE TABLE GRADES(STUD_ID VARCHAR(5), EXAM_ID VARCHAR(5), PRIMARY KEY(STUD_ID, EXAM_ID), FOREIGN KEY (EXAM_ID) REFERENCES EXAM,
FOREIGN KEY (STUD_ID) REFERENCES STUDENT, MARKS INTEGER, GRADE VARCHAR(2));

CREATE TABLE ACCOUNTS(STUD_ID VARCHAR(5), FEE INTEGER, STATUS VARCHAR(3), FOREIGN KEY (STUD_ID) REFERENCES STUDENT);



- **Collect data manually and create Database /Insert the values**

INSERT INTO UNIVERSITY VALUES('B01', 'GITAM, VSKP', 'VISAKHAPATNAM');
INSERT INTO UNIVERSITY VALUES('B02', 'GITAM, BANGLORE', 'BANGLORE');
INSERT INTO UNIVERSITY VALUES('B03', 'GITAM, HYD', 'HYDERABAD');
SELECT * FROM UNIVERSITY;

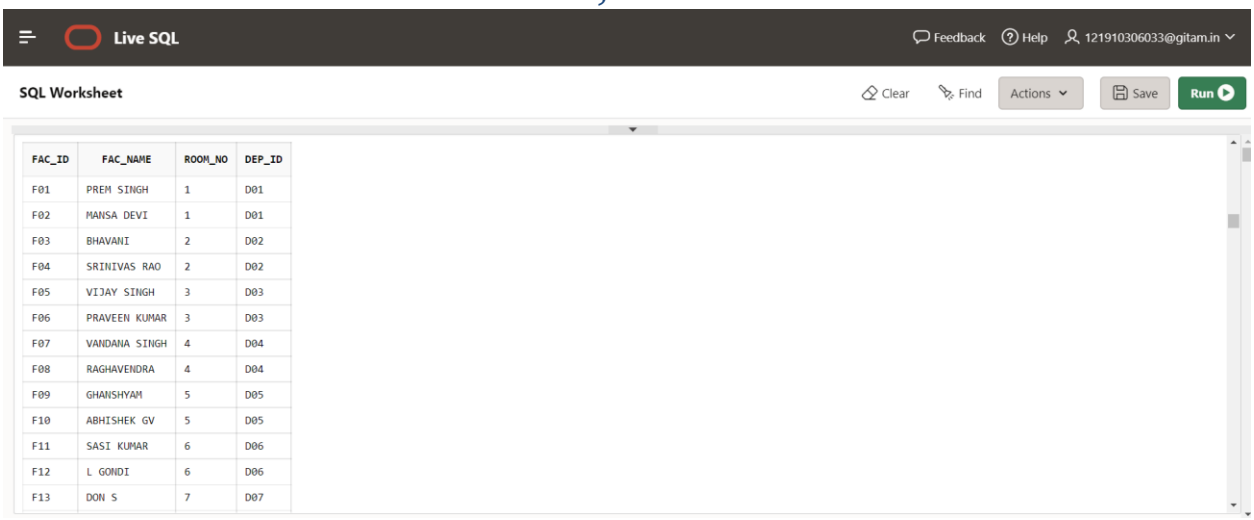


```
INSERT INTO DEPARTMENT VALUES('D01', 'CSE');
INSERT INTO DEPARTMENT VALUES('D02',
'MECH');
INSERT INTO DEPARTMENT VALUES('D03',
'DENTAL');
INSERT INTO DEPARTMENT VALUES('D04', 'LAW');
INSERT INTO DEPARTMENT VALUES('D05', 'ECE');
INSERT INTO DEPARTMENT VALUES('D06',
'PHARMACY');
INSERT INTO DEPARTMENT VALUES('D07', 'EEE');
INSERT INTO DEPARTMENT VALUES('D08',
'MEDICAL');
INSERT INTO DEPARTMENT VALUES('D09', 'ARCH');
SELECT * FROM DEPARTMENT;
```


```
INSERT INTO FACULTY VALUES('F01', 'PREM
SINGH', '1', 'D01');
INSERT INTO FACULTY VALUES('F02', 'MANSA
DEVI', '1', 'D01');
INSERT INTO FACULTY VALUES('F03', 'BHAVANI',
'2', 'D02');
```

```
INSERT INTO FACULTY VALUES('F04', 'SRINIVAS
RAO', '2', 'D02');
INSERT INTO FACULTY VALUES('F05', 'VIJAY
SINGH', '3', 'D03');
INSERT INTO FACULTY VALUES('F06', 'PRAVEEN
KUMAR', '3', 'D03');
INSERT INTO FACULTY VALUES('F07', 'VANDANA
SINGH', '4', 'D04');
INSERT INTO FACULTY VALUES('F08',
'RAGHAVENDRA', '4', 'D04');
INSERT INTO FACULTY VALUES('F09',
'GHANSHYAM', '5', 'D05');
INSERT INTO FACULTY VALUES('F10', 'ABHISHEK
GV', '5', 'D05');
INSERT INTO FACULTY VALUES('F11', 'SASI
KUMAR', '6', 'D06');
INSERT INTO FACULTY VALUES('F12', 'L GONDI', '6',
'D06');
INSERT INTO FACULTY VALUES('F13', 'DON S', '7',
'D07');
INSERT INTO FACULTY VALUES('F14', 'RAVI', '7',
'D07');
INSERT INTO FACULTY VALUES('F15', 'GRACE', '8',
'D08');
INSERT INTO FACULTY VALUES('F16', 'K KAVITA',
'8', 'D08');
INSERT INTO FACULTY VALUES('F17',
'ANKIREDDY', '9', 'D09');
INSERT INTO FACULTY VALUES('F18', 'MURALI
MOHAN', '9', 'D09');
```

```
INSERT INTO FACULTY VALUES('F19',
'SUKHIBHAVA', '2', 'D02');
INSERT INTO FACULTY VALUES('F20',
'TANGUTOORI', '3', 'D03');
INSERT INTO FACULTY VALUES('F21', 'JAYA SRI',
'8', 'D08');
INSERT INTO FACULTY VALUES('F22', 'HIMAJA', '4',
'D04');
INSERT INTO FACULTY VALUES('F23', 'LUCKY', '9',
'D09');
SELECT * FROM FACULTY;
```



FAC_ID	FAC_NAME	ROOM_NO	DEP_ID
F01	PREM SINGH	1	D01
F02	MANSA DEVI	1	D01
F03	BHAVANI	2	D02
F04	SRINIVAS RAO	2	D02
F05	VIJAY SINGH	3	D03
F06	PRAVEEN KUMAR	3	D03
F07	VANDANA SINGH	4	D04
F08	RAGHAVENDRA	4	D04
F09	GHANSHYAM	5	D05
F10	ABHISHEK GV	5	D05
F11	SASI KUMAR	6	D06
F12	L GONDI	6	D06
F13	DON S	7	D07

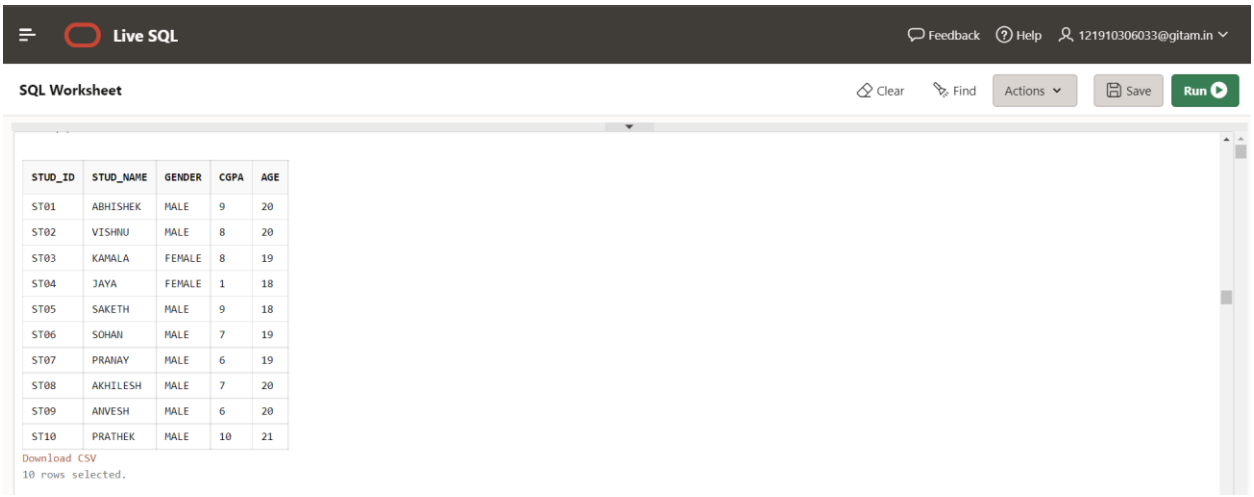
```
INSERT INTO SUBJECT VALUES('S01', 'ADV DATA
STRUCTURES', 'F01', 'D01');
INSERT INTO SUBJECT VALUES('S02', 'DBMS', 'F02',
'D01');
INSERT INTO SUBJECT VALUES('S03', 'THERMO
DYNAMICS', 'F03', 'D02');
INSERT INTO SUBJECT VALUES('S04', 'PHYSICS',
'F04', 'D02');
INSERT INTO SUBJECT VALUES('S05', 'TEETH', 'F05',
'D03');
```

```
INSERT INTO SUBJECT VALUES('S06', 'GUMS', 'F06',
'D03');
INSERT INTO SUBJECT VALUES('S07', 'CIVICS', 'F07',
'D04');
INSERT INTO SUBJECT VALUES('S08',
'CRIMINOLOGY', 'F08', 'D04');
INSERT INTO SUBJECT VALUES('S09', 'FDLC', 'F09',
'D05');
INSERT INTO SUBJECT VALUES('S10', 'COA', 'F10',
'D05');
INSERT INTO SUBJECT VALUES('S11', 'LIFE
SCIENCES', 'F11', 'D06');
INSERT INTO SUBJECT VALUES('S12', 'BIO
CHEMISTRY', 'F11', 'D06');
INSERT INTO SUBJECT VALUES('S13', 'BEEE', 'F13',
'D07');
INSERT INTO SUBJECT VALUES('S14',
'ELECTRICAL', 'F14', 'D07');
INSERT INTO SUBJECT VALUES('S15', 'HUMAN
BODY', 'F15', 'D08');
INSERT INTO SUBJECT VALUES('S16',
'REPRODUCTION', 'F16', 'D08');
INSERT INTO SUBJECT VALUES('S17', 'SCALES',
'F17', 'D09');
INSERT INTO SUBJECT VALUES('S18', 'AUTOCAD',
'F18', 'D09');
SELECT * FROM SUBJECT;
```

SUB_ID	SUB_NAME	FAC_ID	DEP_ID
S01	ADV DATA STRUCTURES	F01	D01
S02	DBMS	F02	D01
S03	THERMO DYNAMICS	F03	D02
S04	PHYSICS	F04	D02
S05	TEETH	F05	D03
S06	GUMS	F06	D03
S07	CIVICS	F07	D04
S08	CRIMINOLOGY	F08	D04
S09	FDLC	F09	D05
S10	COA	F10	D05
S11	LIFE SCIENCES	F11	D06
S12	BIO CHEMISTRY	F11	D06
S13	BEEE	F13	D07

```
INSERT INTO STUDENT VALUES('ST01',
'ABHISHEK', 'MALE', 9, 20);
INSERT INTO STUDENT VALUES('ST02', 'VISHNU',
'MALE', 8, 20);
INSERT INTO STUDENT VALUES('ST03', 'KAMALA',
'FEMALE', 8, 19);
INSERT INTO STUDENT VALUES('ST04', 'JAYA',
'FEMALE', 1, 18);
INSERT INTO STUDENT VALUES('ST05', 'SAKETH',
'MALE', 9, 18);
INSERT INTO STUDENT VALUES('ST06', 'SOHAN',
'MALE', 7, 19);
INSERT INTO STUDENT VALUES('ST07', 'PRANAY',
'MALE', 6, 19);
INSERT INTO STUDENT VALUES('ST08',
'AKHILESH', 'MALE', 7, 20);
INSERT INTO STUDENT VALUES('ST09', 'ANVESH',
'MALE', 6, 20);
```

```
INSERT INTO STUDENT VALUES('ST10', 'PRATHEK',
'MALE', 10, 21);
SELECT * FROM STUDENT;
```

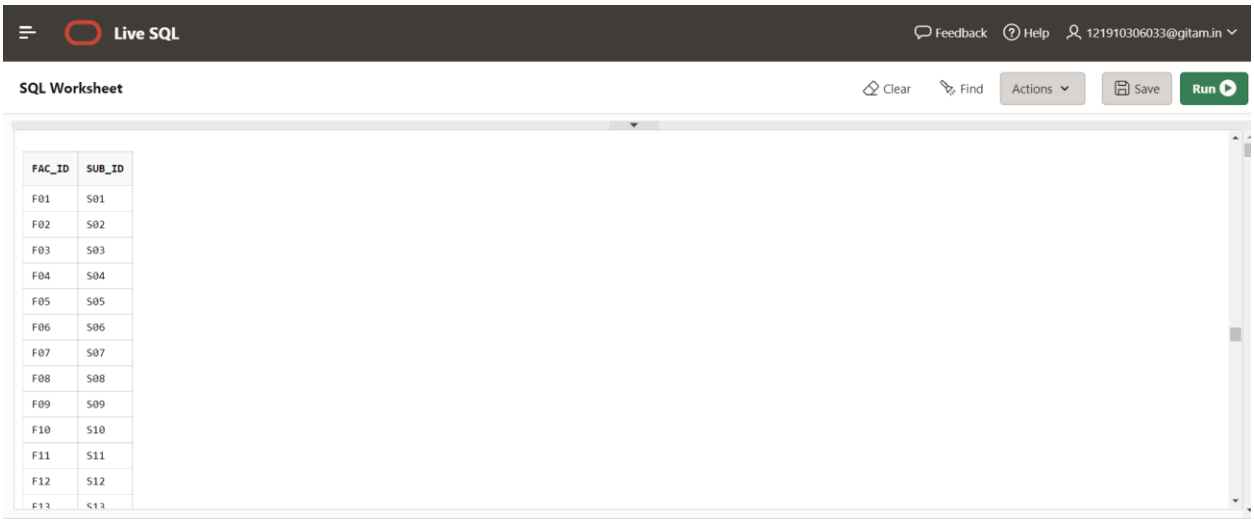


STUD_ID	STUD_NAME	GENDER	CGPA	AGE
ST01	ABHISHEK	MALE	9	20
ST02	VISHNU	MALE	8	20
ST03	KAMALA	FEMALE	8	19
ST04	JAYA	FEMALE	1	18
ST05	SAKETH	MALE	9	18
ST06	SOHAN	MALE	7	19
ST07	PRANAY	MALE	6	19
ST08	AKHILESH	MALE	7	20
ST09	ANVESH	MALE	6	20
ST10	PRATHEK	MALE	10	21

Download CSV
10 rows selected.

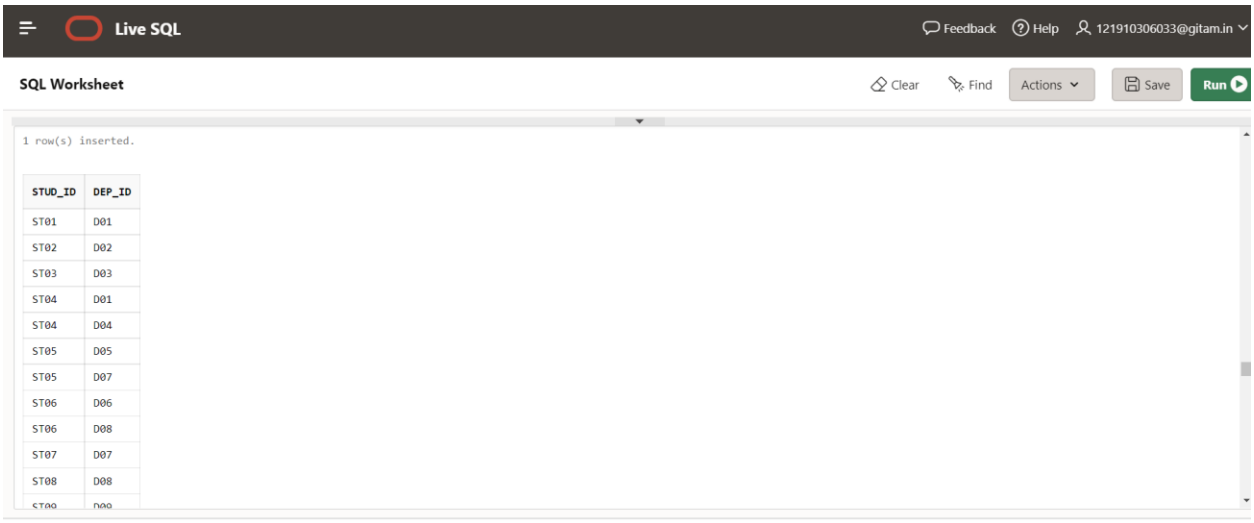
```
INSERT INTO TEACHES VALUES('F01', 'S01');
INSERT INTO TEACHES VALUES('F02', 'S02');
INSERT INTO TEACHES VALUES('F03', 'S03');
INSERT INTO TEACHES VALUES('F04', 'S04');
INSERT INTO TEACHES VALUES('F05', 'S05');
INSERT INTO TEACHES VALUES('F06', 'S06');
INSERT INTO TEACHES VALUES('F07', 'S07');
INSERT INTO TEACHES VALUES('F08', 'S08');
INSERT INTO TEACHES VALUES('F09', 'S09');
INSERT INTO TEACHES VALUES('F10', 'S10');
INSERT INTO TEACHES VALUES('F11', 'S11');
INSERT INTO TEACHES VALUES('F12', 'S12');
INSERT INTO TEACHES VALUES('F13', 'S13');
INSERT INTO TEACHES VALUES('F14', 'S14');
INSERT INTO TEACHES VALUES('F15', 'S15');
INSERT INTO TEACHES VALUES('F16', 'S16');
INSERT INTO TEACHES VALUES('F17', 'S17');
```

```
INSERT INTO TEACHES VALUES('F18', 'S18');
SELECT* FROM TEACHES;
```

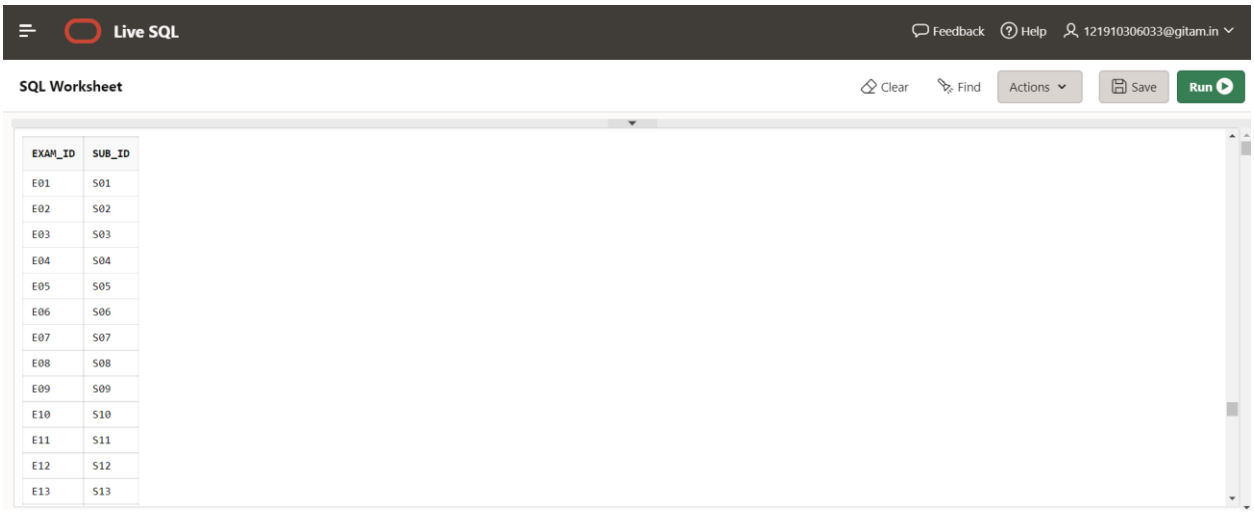


FAC_ID	SUB_ID
F01	S01
F02	S02
F03	S03
F04	S04
F05	S05
F06	S06
F07	S07
F08	S08
F09	S09
F10	S10
F11	S11
F12	S12
F13	S13

```
INSERT INTO ENROLLED VALUES('ST01', 'D01');
INSERT INTO ENROLLED VALUES('ST02', 'D02');
INSERT INTO ENROLLED VALUES('ST03', 'D03');
INSERT INTO ENROLLED VALUES('ST04', 'D04');
INSERT INTO ENROLLED VALUES('ST05', 'D05');
INSERT INTO ENROLLED VALUES('ST06', 'D06');
INSERT INTO ENROLLED VALUES('ST07', 'D07');
INSERT INTO ENROLLED VALUES('ST08', 'D08');
INSERT INTO ENROLLED VALUES('ST09', 'D09');
INSERT INTO ENROLLED VALUES('ST10', 'D01');
INSERT INTO ENROLLED VALUES('ST04', 'D01');
INSERT INTO ENROLLED VALUES('ST05', 'D07');
INSERT INTO ENROLLED VALUES('ST06', 'D08');
SELECT* FROM ENROLLED;
```

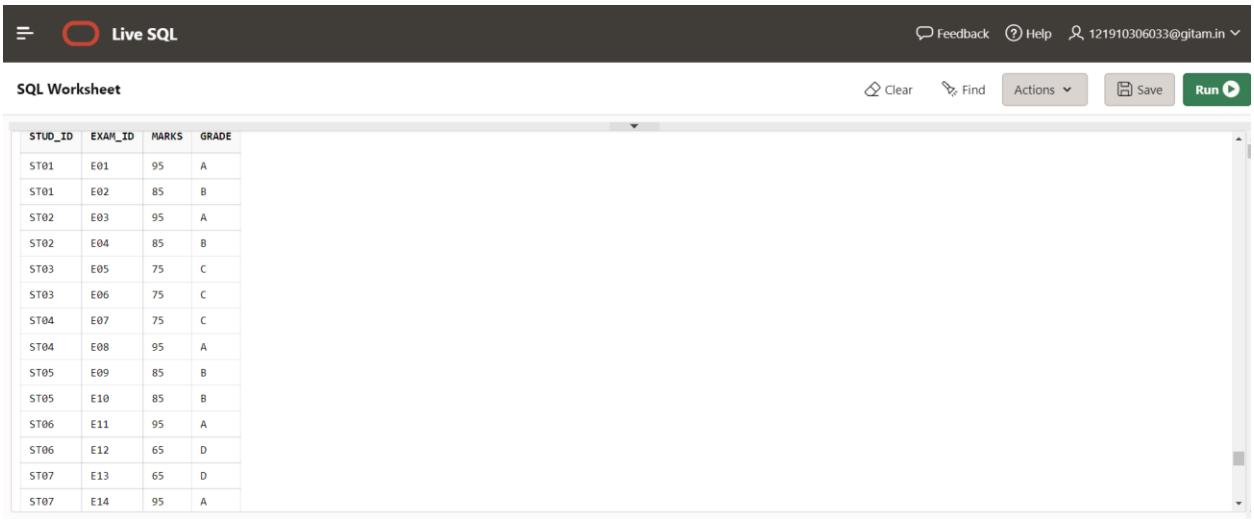



```
INSERT INTO EXAM VALUES('E01', 'S01');
INSERT INTO EXAM VALUES('E02', 'S02');
INSERT INTO EXAM VALUES('E03', 'S03');
INSERT INTO EXAM VALUES('E04', 'S04');
INSERT INTO EXAM VALUES('E05', 'S05');
INSERT INTO EXAM VALUES('E06', 'S06');
INSERT INTO EXAM VALUES('E07', 'S07');
INSERT INTO EXAM VALUES('E08', 'S08');
INSERT INTO EXAM VALUES('E09', 'S09');
INSERT INTO EXAM VALUES('E10', 'S10');
INSERT INTO EXAM VALUES('E11', 'S11');
INSERT INTO EXAM VALUES('E12', 'S12');
INSERT INTO EXAM VALUES('E13', 'S13');
INSERT INTO EXAM VALUES('E14', 'S14');
INSERT INTO EXAM VALUES('E15', 'S15');
INSERT INTO EXAM VALUES('E16', 'S16');
INSERT INTO EXAM VALUES('E17', 'S17');
INSERT INTO EXAM VALUES('E18', 'S18');
SELECT * FROM EXAM;
```



EXAM_ID	SUB_ID
E01	S01
E02	S02
E03	S03
E04	S04
E05	S05
E06	S06
E07	S07
E08	S08
E09	S09
E10	S10
E11	S11
E12	S12
E13	S13

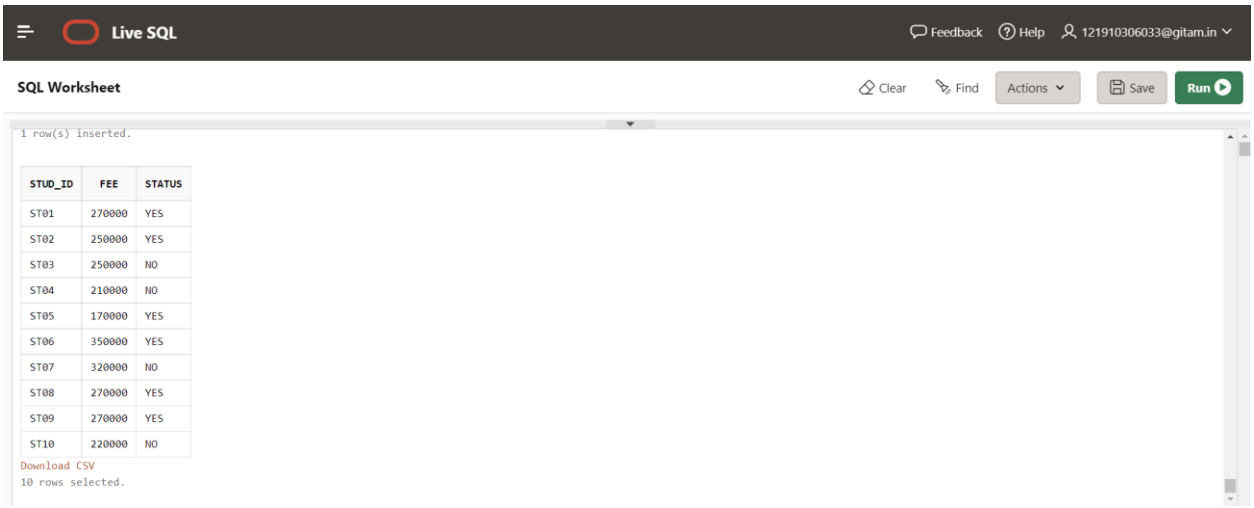
```
INSERT INTO GRADES VALUES('ST01', 'E01' ,95, 'A');
INSERT INTO GRADES VALUES('ST01', 'E02' ,85, 'B');
INSERT INTO GRADES VALUES('ST02', 'E03' ,95, 'A');
INSERT INTO GRADES VALUES('ST02', 'E04' ,85, 'B');
INSERT INTO GRADES VALUES('ST03', 'E05' ,75, 'C');
INSERT INTO GRADES VALUES('ST03', 'E06' ,75, 'C');
INSERT INTO GRADES VALUES('ST04', 'E07' ,75, 'C');
INSERT INTO GRADES VALUES('ST04', 'E08' ,95, 'A');
INSERT INTO GRADES VALUES('ST05', 'E09' ,85, 'B');
INSERT INTO GRADES VALUES('ST05', 'E10' ,85, 'B');
INSERT INTO GRADES VALUES('ST06', 'E11' ,95, 'A');
INSERT INTO GRADES VALUES('ST06', 'E12' ,65, 'D');
INSERT INTO GRADES VALUES('ST07', 'E13' ,65, 'D');
INSERT INTO GRADES VALUES('ST07', 'E14' ,95, 'A');
INSERT INTO GRADES VALUES('ST08', 'E15' ,75, 'C');
INSERT INTO GRADES VALUES('ST08', 'E16' ,85, 'B');
INSERT INTO GRADES VALUES('ST09', 'E17' ,85, 'B');
INSERT INTO GRADES VALUES('ST10', 'E18' ,95, 'A');
SELECT * FROM GRADES;
```



STUD_ID	EXAM_ID	MARKS	GRADE
ST01	E01	95	A
ST01	E02	85	B
ST02	E03	95	A
ST02	E04	85	B
ST03	E05	75	C
ST03	E06	75	C
ST04	E07	75	C
ST04	E08	95	A
ST05	E09	85	B
ST05	E10	85	B
ST06	E11	95	A
ST06	E12	65	D
ST07	E13	65	D
ST07	E14	95	A

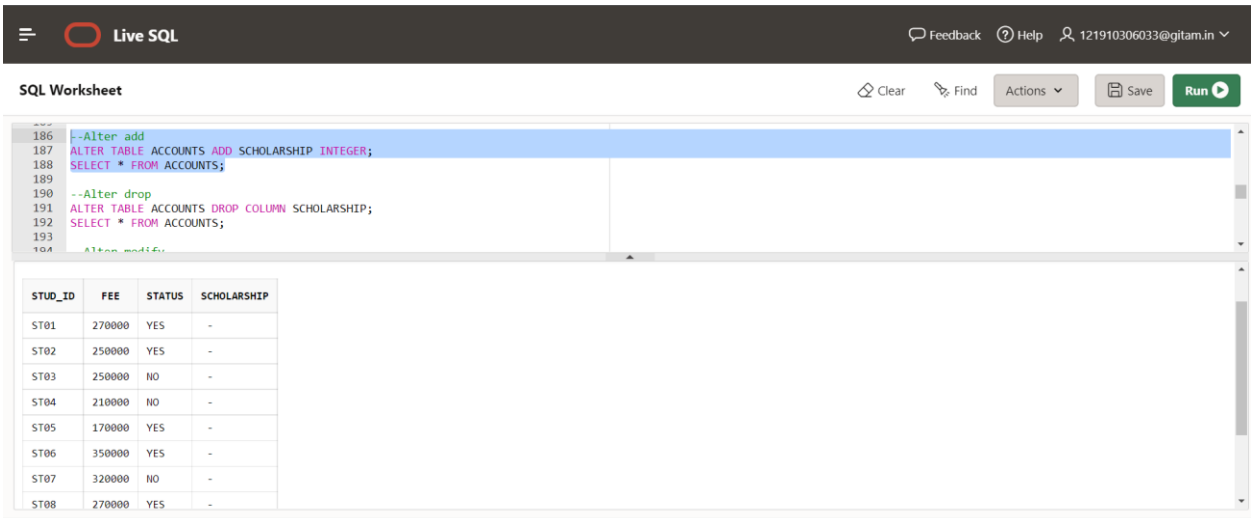
```
INSERT INTO ACCOUNTS VALUES('ST01', 270000,
'YES');
INSERT INTO ACCOUNTS VALUES('ST02', 250000,
'YES');
INSERT INTO ACCOUNTS VALUES('ST03', 250000,
'NO');
INSERT INTO ACCOUNTS VALUES('ST04', 210000,
'NO');
INSERT INTO ACCOUNTS VALUES('ST05', 170000,
'YES');
INSERT INTO ACCOUNTS VALUES('ST06', 350000,
'YES');
INSERT INTO ACCOUNTS VALUES('ST07', 320000,
'NO');
INSERT INTO ACCOUNTS VALUES('ST08', 270000,
'YES');
INSERT INTO ACCOUNTS VALUES('ST09', 270000,
'YES');
INSERT INTO ACCOUNTS VALUES('ST10', 220000,
'NO');
```

SELECT * FROM ACCOUNTS;



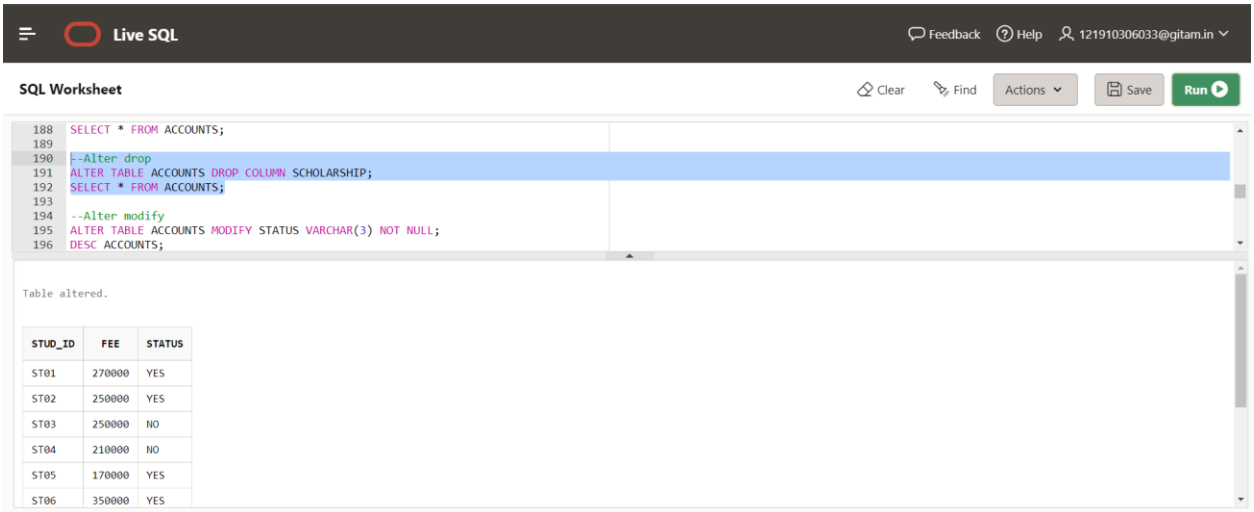
- **Perform all the Various SQL Commands Operations.**

--ALTER ADD
ALTER TABLE ACCOUNTS ADD SCHOLARSHIP INTEGER;
SELECT * FROM ACCOUNTS;



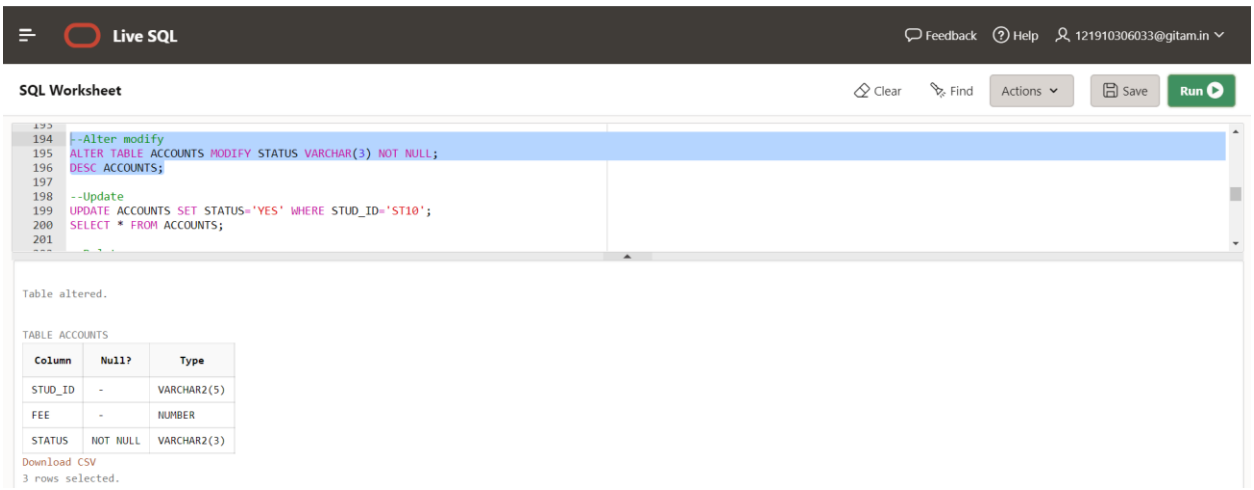
--ALTER DROP

```
ALTER TABLE ACCOUNTS DROP COLUMN SCHOLARSHIP;
SELECT * FROM ACCOUNTS;
```



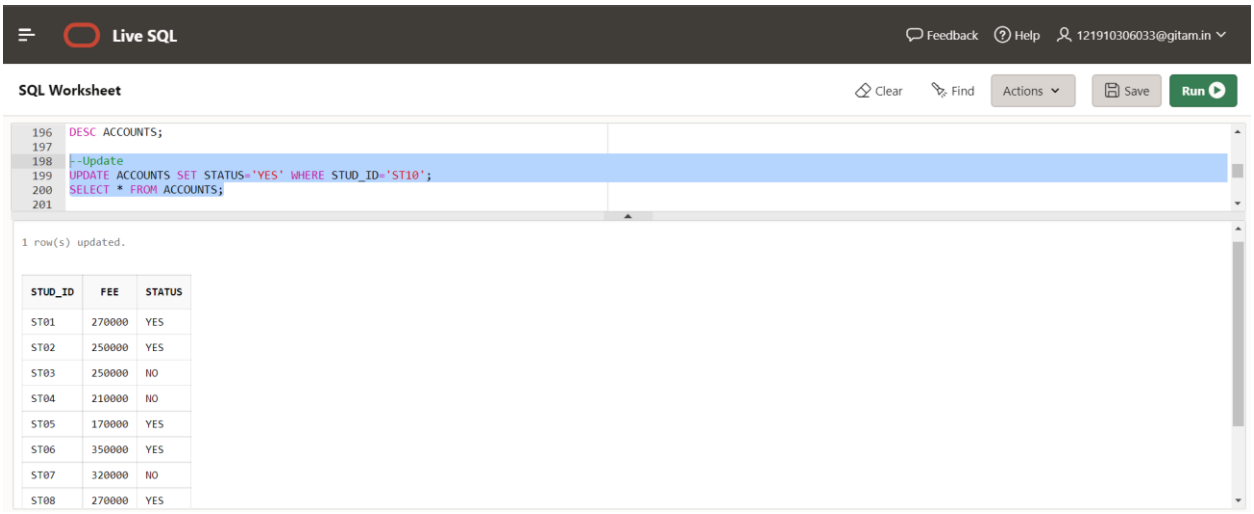
--ALTER MODIFY

```
ALTER TABLE ACCOUNTS MODIFY STATUS VARCHAR(3) NOT NULL;
DESC ACCOUNTS;
```



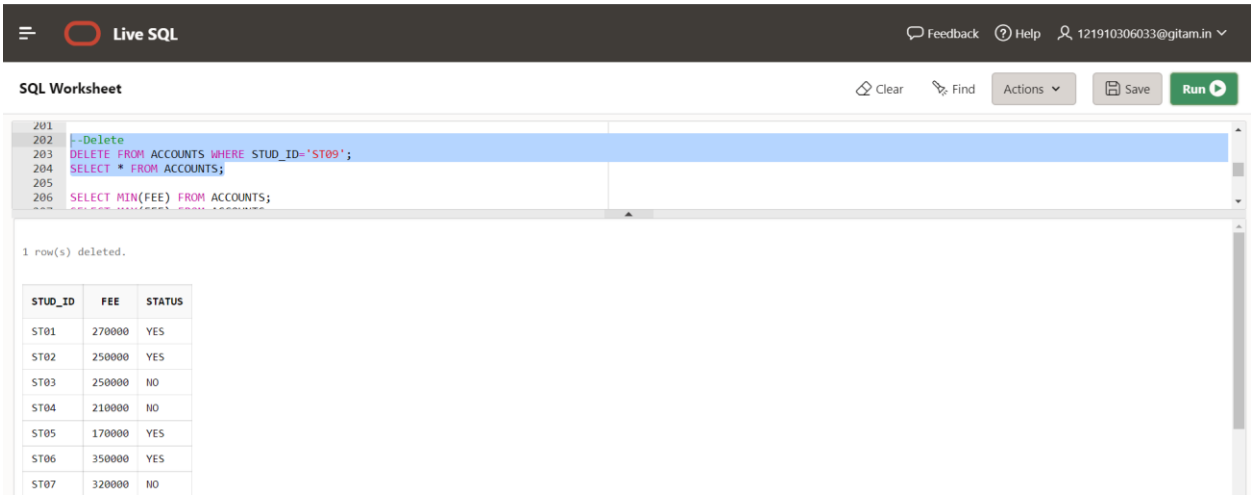
--UPDATE

```
UPDATE ACCOUNTS SET STATUS='YES' WHERE
STUD_ID='ST10';
SELECT * FROM ACCOUNTS;
```



--DELETE

```
DELETE FROM ACCOUNTS WHERE
STUD_ID='ST09';
SELECT * FROM ACCOUNTS;
```

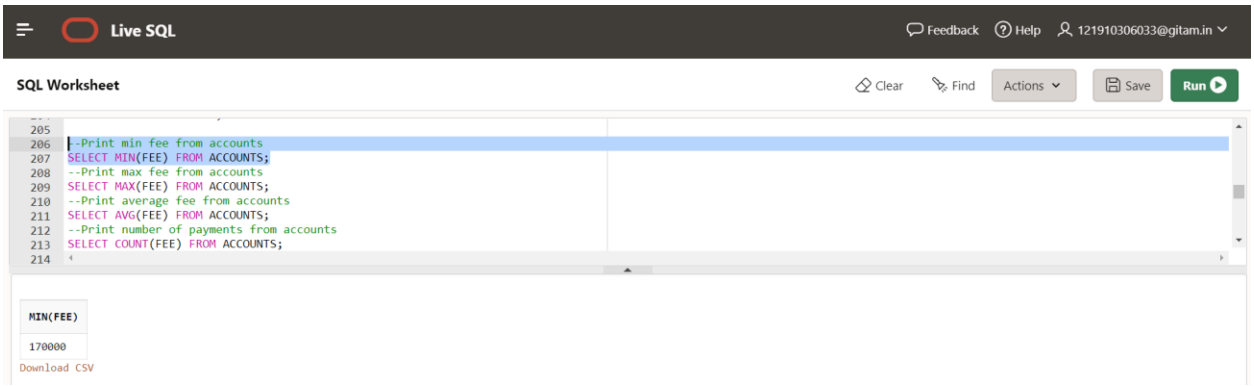


Part-IV Queries:

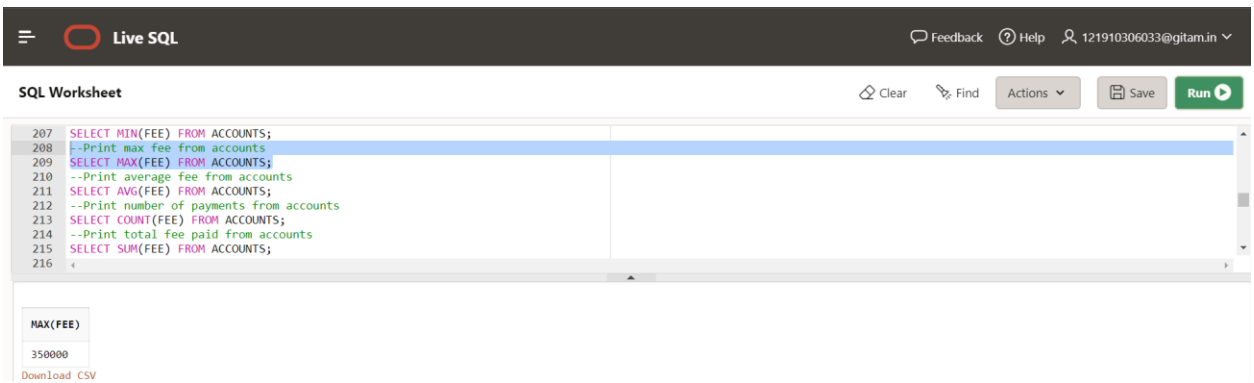
Write Queries Related To Database

1. Aggregate Operators

```
--PRINT MIN FEE FROM ACCOUNTS
SELECT MIN(FEE) FROM ACCOUNTS;
```

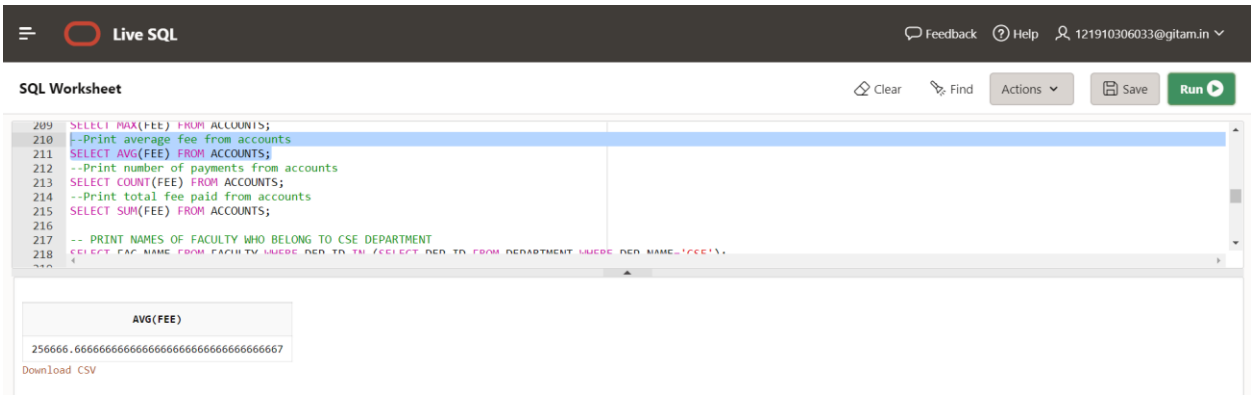



```
--PRINT MAX FEE FROM ACCOUNTS
SELECT MAX(FEE) FROM ACCOUNTS;
```



--PRINT AVERAGE FEE FROM ACCOUNTS

SELECT AVG(FEE) FROM ACCOUNTS;

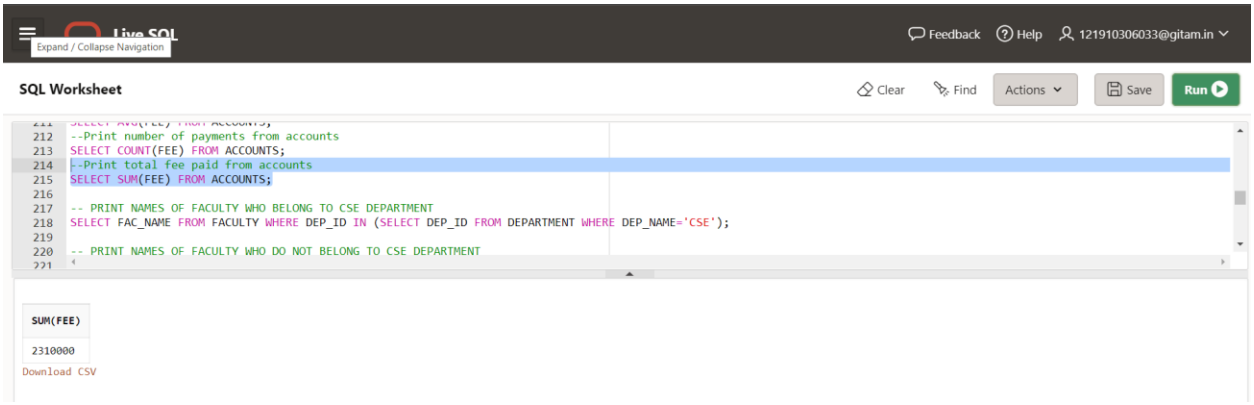


--PRINT NUMBER OF PAYMENTS FROM ACCOUNTS

SELECT COUNT(FEE) FROM ACCOUNTS;

--PRINT TOTAL FEE PAID FROM ACCOUNTS

SELECT SUM(FEE) FROM ACCOUNTS;

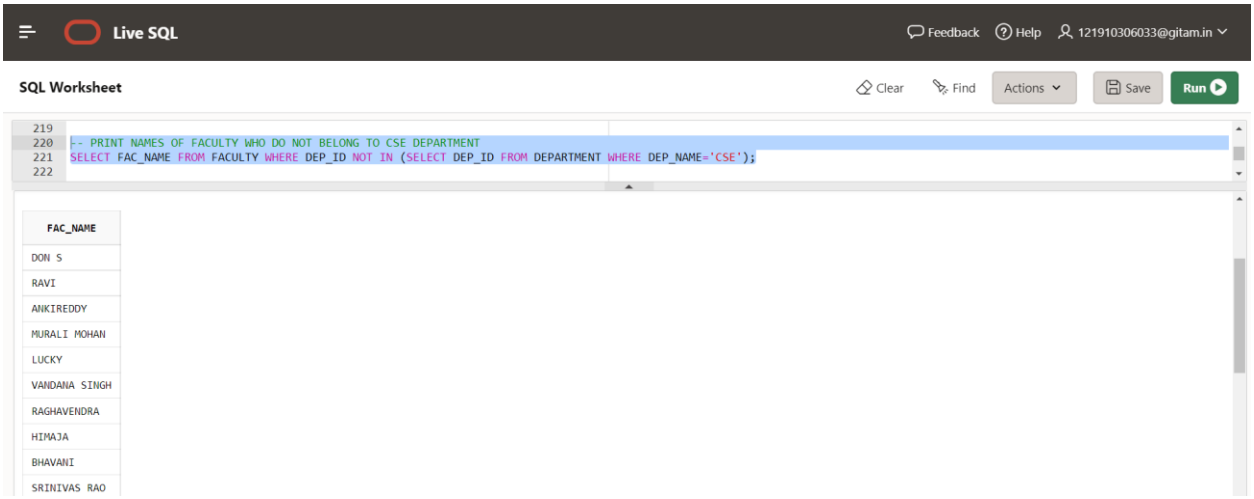


2. Nested Queries

-- PRINT NAMES OF FACULTY WHO BELONG TO CSE DEPARTMENT

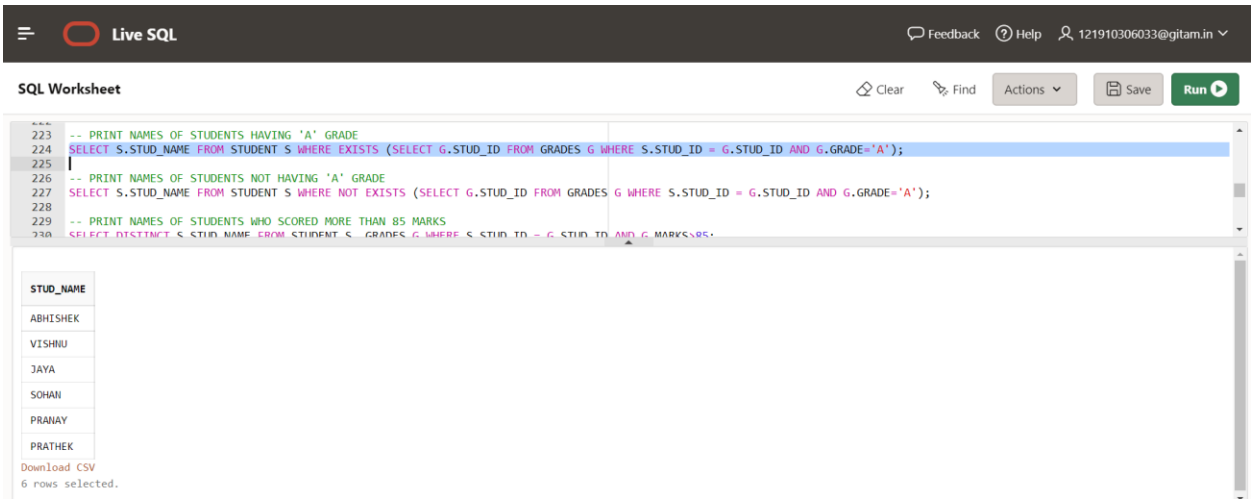
SELECT FAC_NAME FROM FACULTY WHERE DEP_ID IN (SELECT DEP_ID FROM DEPARTMENT WHERE DEP_NAME='CSE');

```
-- PRINT NAMES OF FACULTY WHO DO NOT BELONG TO CSE DEPARTMENT
SELECT FAC_NAME FROM FACULTY WHERE DEP_ID NOT IN (SELECT DEP_ID FROM DEPARTMENT WHERE DEP_NAME='CSE');
```



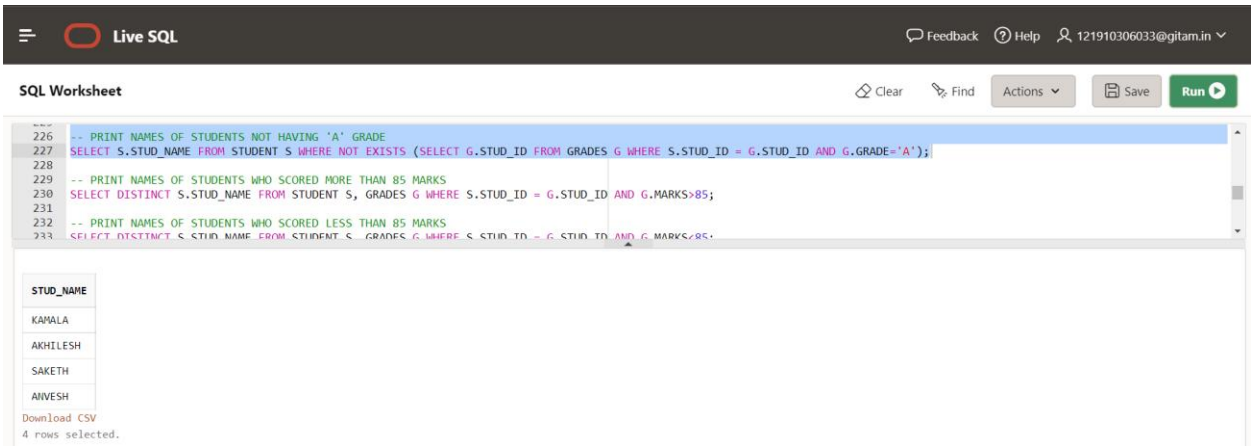
3. Correlated Nested Query

```
-- PRINT NAMES OF STUDENTS HAVING 'A' GRADE
SELECT S.STUD_NAME FROM STUDENT S WHERE EXISTS (SELECT G.STUD_ID FROM GRADES G WHERE S.STUD_ID = G.STUD_ID AND G.GRADE='A');
```



-- PRINT NAMES OF STUDENTS NOT HAVING 'A' GRADE

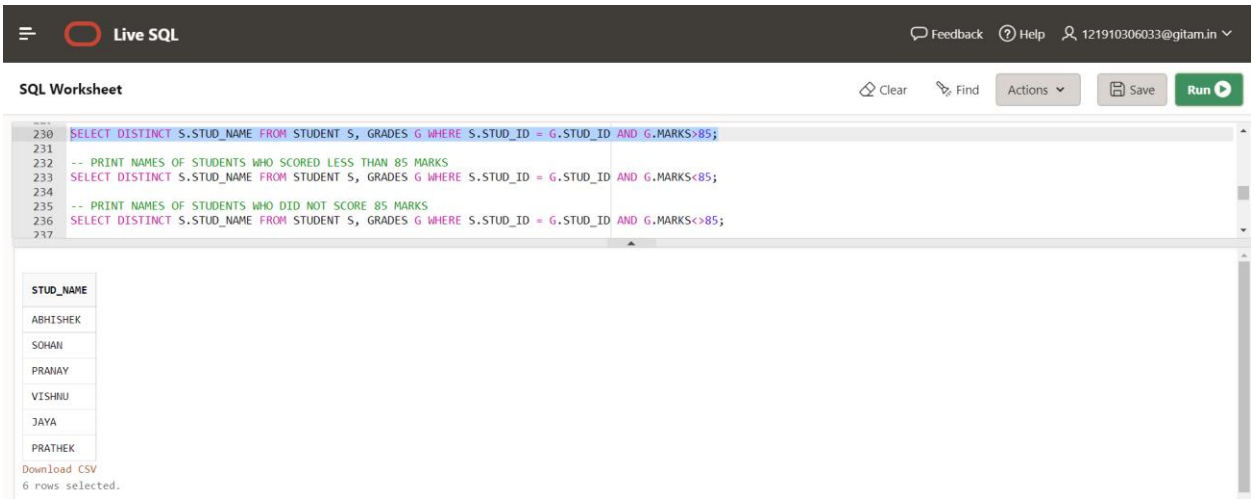
SELECT S.STUD_NAME FROM STUDENT S WHERE NOT EXISTS (SELECT G.STUD_ID FROM GRADES G WHERE S.STUD_ID = G.STUD_ID AND G.GRADE='A');



4. Set Comparison Operators

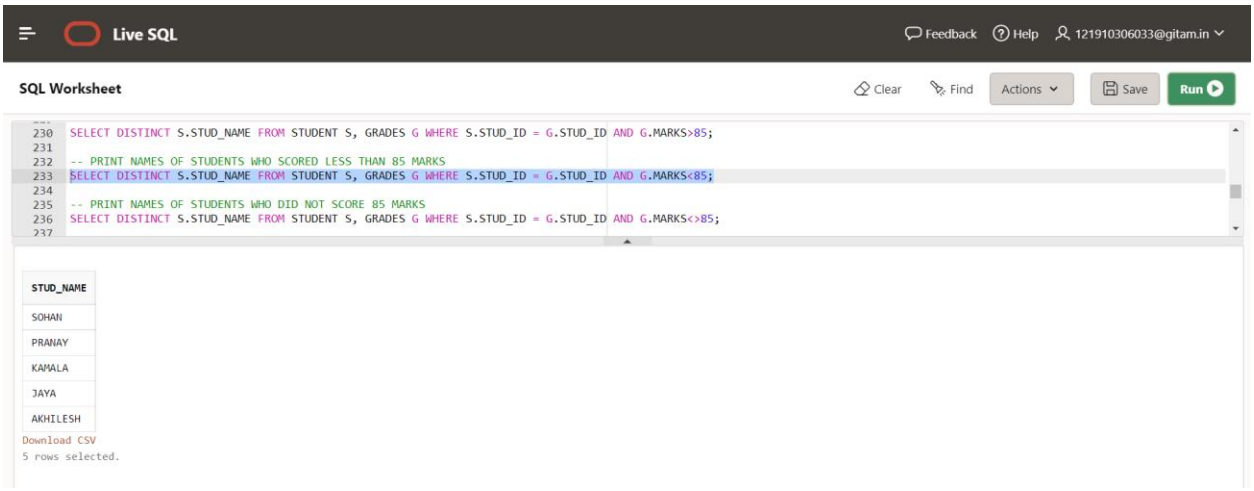
-- PRINT NAMES OF STUDENTS WHO SCORED MORE THAN 85 MARKS

SELECT DISTINCT S.STUD_NAME FROM STUDENT S, GRADES G WHERE S.STUD_ID = G.STUD_ID AND G.MARKS>85;



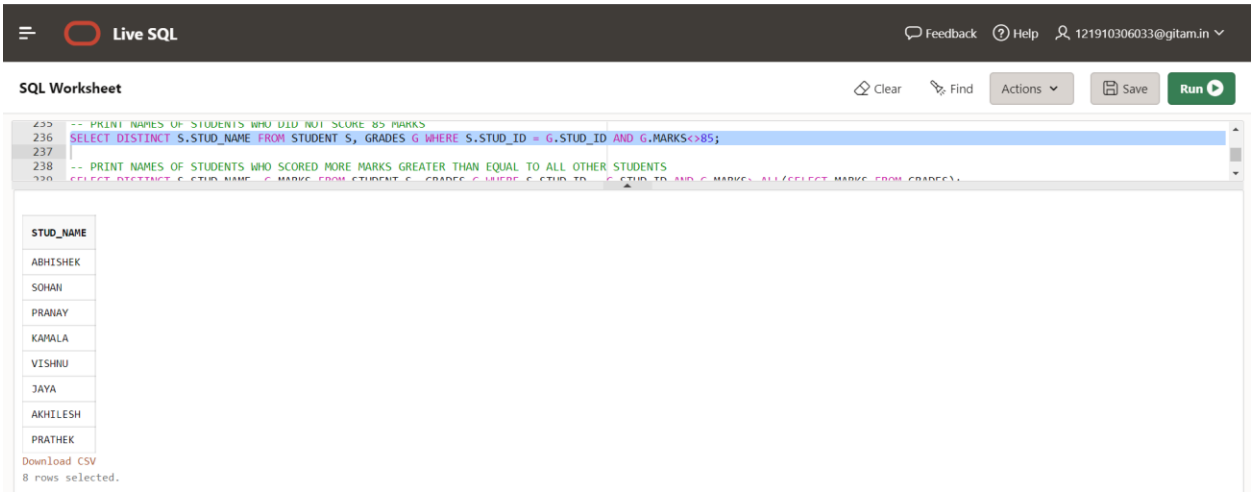
-- PRINT NAMES OF STUDENTS WHO SCORED LESS THAN 85 MARKS

SELECT DISTINCT S.STUD_NAME FROM STUDENT S, GRADES G WHERE S.STUD_ID = G.STUD_ID AND G.MARKS<85;



-- PRINT NAMES OF STUDENTS WHO DID NOT SCORE 85 MARKS

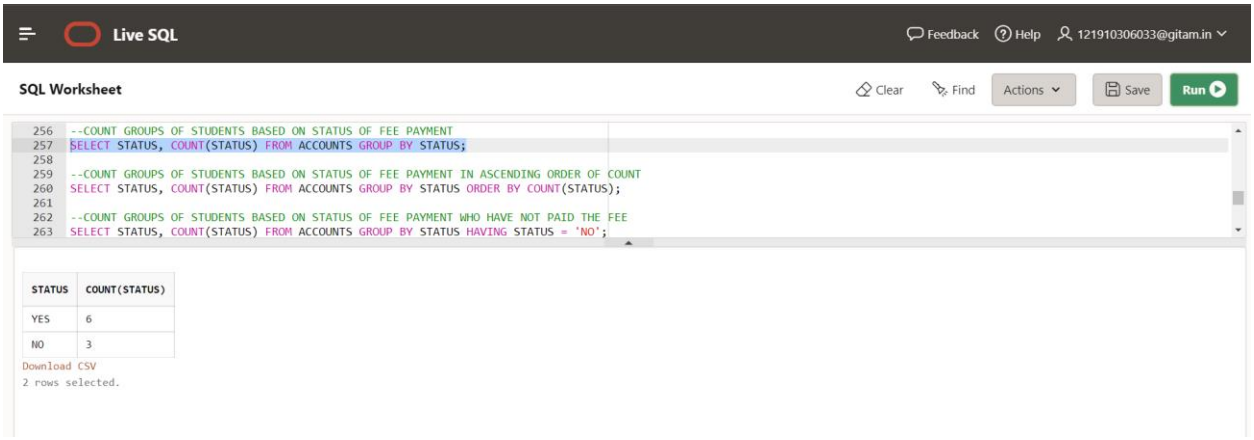
SELECT DISTINCT S.STUD_NAME FROM STUDENT S, GRADES G WHERE S.STUD_ID = G.STUD_ID AND G.MARKS<>85;



5. Groupby Having

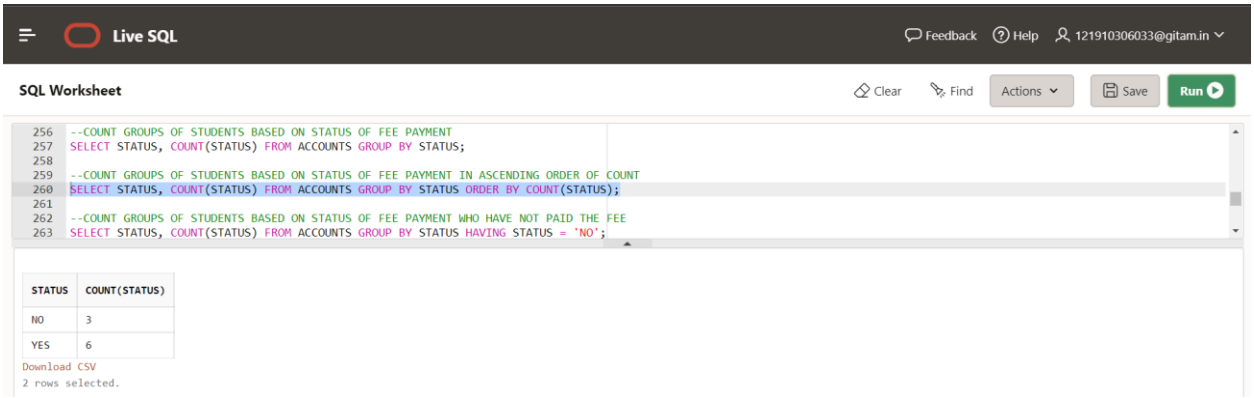
--COUNT GROUPS OF STUDENTS BASED ON STATUS OF FEE PAYMENT

SELECT STATUS, COUNT(STATUS) FROM ACCOUNTS GROUP BY STATUS;



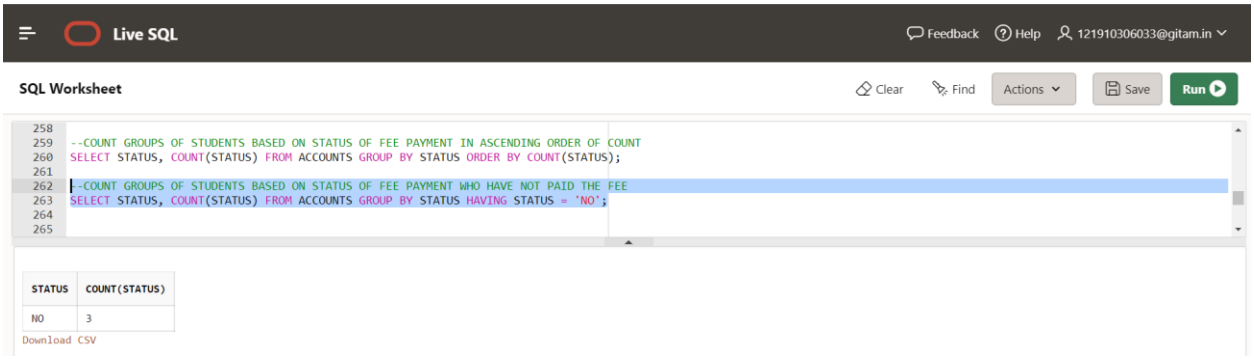
--COUNT GROUPS OF STUDENTS BASED ON STATUS OF FEE PAYMENT IN ASCENDING ORDER OF COUNT

SELECT STATUS, COUNT(STATUS) FROM ACCOUNTS GROUP BY STATUS ORDER BY COUNT(STATUS);



--COUNT GROUPS OF STUDENTS BASED ON STATUS OF FEE PAYMENT WHO HAVE NOT PAID THE FEE

SELECT STATUS, COUNT(STATUS) FROM ACCOUNTS GROUP BY STATUS HAVING STATUS = 'NO';



6. Relational SET Queries

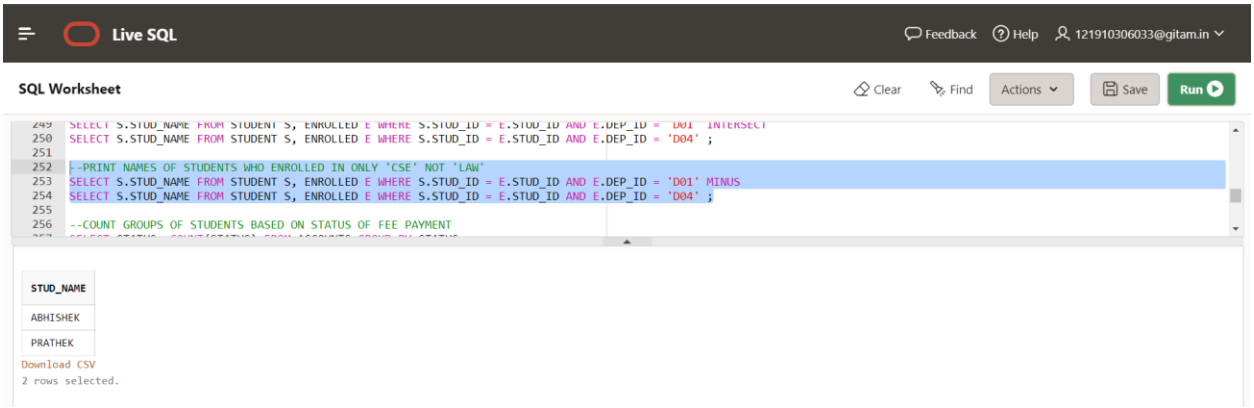
```
--PRINT NAMES OF STUDENTS WHO ENROLLED IN EITHER 'CSE' OR 'EEE'
SELECT S.STUD_NAME FROM STUDENT S, ENROLLED E WHERE S.STUD_ID = E.STUD_ID AND E.DEP_ID = 'D01' UNION
SELECT S.STUD_NAME FROM STUDENT S, ENROLLED E WHERE S.STUD_ID = E.STUD_ID AND E.DEP_ID = 'D07' ;
```


```
--PRINT NAMES OF STUDENTS WHO ENROLLED IN BOTH 'CSE' AND 'LAW'
SELECT S.STUD_NAME FROM STUDENT S, ENROLLED E WHERE S.STUD_ID = E.STUD_ID AND E.DEP_ID = 'D01' INTERSECT
SELECT S.STUD_NAME FROM STUDENT S, ENROLLED E WHERE S.STUD_ID = E.STUD_ID AND E.DEP_ID = 'D04' ;
```

--PRINT NAMES OF STUDENTS WHO ENROLLED IN ONLY 'CSE' NOT 'LAW'

SELECT S.STUD_NAME FROM STUDENT S, ENROLLED E WHERE S.STUD_ID = E.STUD_ID AND E.DEP_ID = 'D01' MINUS

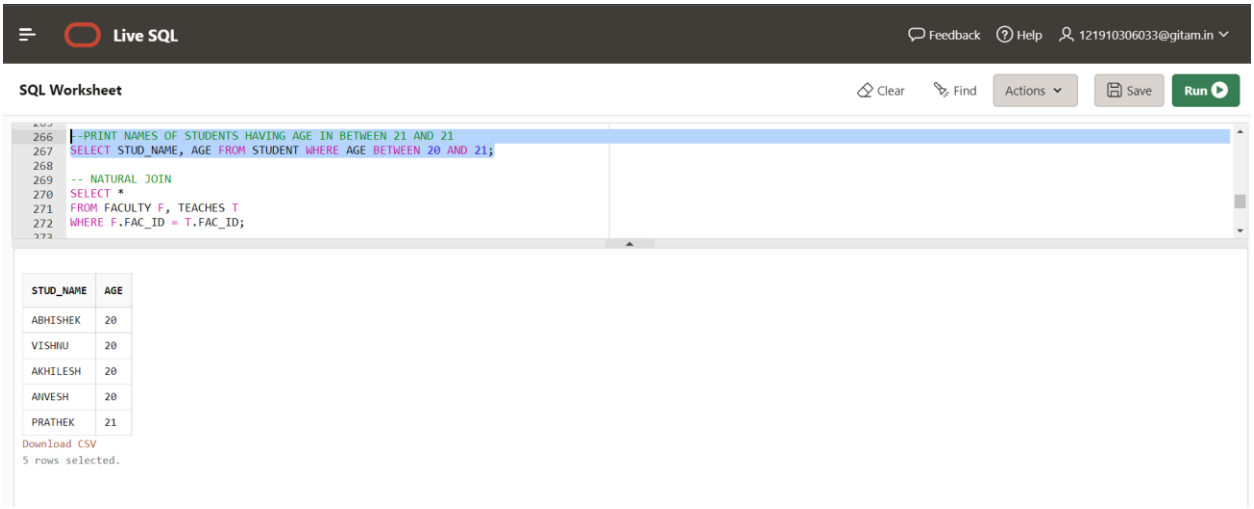
SELECT S.STUD_NAME FROM STUDENT S, ENROLLED E WHERE S.STUD_ID = E.STUD_ID AND E.DEP_ID = 'D04' ;



7. Between, Like operator

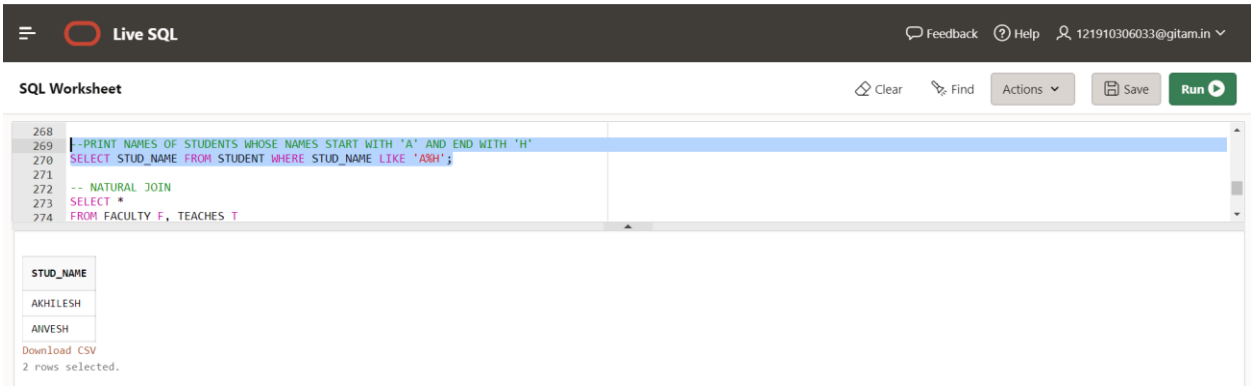
--PRINT NAMES OF STUDENTS HAVING AGE IN BETWEEN 21 AND 21

SELECT STUD_NAME, AGE FROM STUDENT WHERE AGE BETWEEN 20 AND 21;



--PRINT NAMES OF STUDENTS WHOSE NAMES START WITH 'A' AND END WITH 'H'

SELECT STUD_NAME FROM STUDENT WHERE STUD_NAME LIKE 'A%H';


Live SQL
Feedback
Help
121910306033@gitam.in
SQL Worksheet
Clear
Find
Actions
Save
Run
268
269
270
271
272
273
274
--PRINT NAMES OF STUDENTS WHOSE NAMES START WITH 'A' AND END WITH 'H'
SELECT STUD_NAME FROM STUDENT WHERE STUD_NAME LIKE 'A%H';
-- NATURAL JOIN
SELECT *
FROM FACULTY F, TEACHES T
STUD_NAME
AKHILESH
ANVESH
Download CSV
2 rows selected.

8. Joins

```
-- NATURAL JOIN
SELECT *
FROM FACULTY F, TEACHES T
WHERE F.FAC_ID = T.FAC_ID;
```

FAC_ID	FAC_NAME	ROOM_NO	DEP_ID	FAC_ID	SUB_ID
F01	PREM SINGH	1	D01	F01	S01
F02	MANSA DEVI	1	D01	F02	S02
F03	BHAVANI	2	D02	F03	S03
F04	SRINIVAS RAO	2	D02	F04	S04
F05	VIJAY SINGH	3	D03	F05	S05
F06	PRAVEEN KUMAR	3	D03	F06	S06
F07	VANDANA SINGH	4	D04	F07	S07
F08	RAGHAVENDRA	4	D04	F08	S08
F09	GHANSHYAM	5	D05	F09	S09
F10	ABHISHEK GV	5	D05	F10	S10
F11	SASI KUMAR	6	D06	F11	S11
F12	L GONDI	6	D06	F12	S12
F13	DON S	7	D07	F13	S13
F14	RAVI	7	D07	F14	S14
F15	GRACE	8	D08	F15	S15
F16	K KAVITA	8	D08	F16	S16
F17	ANKIREDDY	9	D09	F17	S17
F18	MURALI MOHAN	9	D09	F18	S18

```
-- INNER JOIN
SELECT *
FROM FACULTY F
INNER JOIN TEACHES T
ON F.FAC_ID = T.FAC_ID;
```

FAC_ID	FAC_NAME	ROOM_NO	DEP_ID	FAC_ID	SUB_ID
F01	PREM SINGH	1	D01	F01	S01
F02	MANSA DEVI	1	D01	F02	S02
F03	BHAVANI	2	D02	F03	S03
F04	SRINIVAS RAO	2	D02	F04	S04
F05	VIJAY SINGH	3	D03	F05	S05
F06	PRAVEEN KUMAR	3	D03	F06	S06
F07	VANDANA SINGH	4	D04	F07	S07
F08	RAGHAVENDRA	4	D04	F08	S08
F09	GHANSHYAM	5	D05	F09	S09
F10	ABHISHEK GV	5	D05	F10	S10
F11	SASI KUMAR	6	D06	F11	S11
F12	L GONDI	6	D06	F12	S12
F13	DON S	7	D07	F13	S13
F14	RAVI	7	D07	F14	S14
F15	GRACE	8	D08	F15	S15
F16	K KAVITA	8	D08	F16	S16
F17	ANKIREDDY	9	D09	F17	S17
F18	MURALI MOHAN	9	D09	F18	S18

```
-- LEFT JOIN
SELECT *
FROM FACULTY F
LEFT JOIN TEACHES T
ON F.FAC_ID = T.FAC_ID;
```

FAC_ID	FAC_NAME	ROOM_NO	DEP_ID	FAC_ID	SUB_ID
F01	PREM SINGH	1	D01	F01	S01
F02	MANSA DEVI	1	D01	F02	S02
F03	BHAVANI	2	D02	F03	S03
F04	SRINIVAS RAO	2	D02	F04	S04
F05	VIJAY SINGH	3	D03	F05	S05
F06	PRAVEEN KUMAR	3	D03	F06	S06
F07	VANDANA SINGH	4	D04	F07	S07
F08	RAGHAVENDRA	4	D04	F08	S08
F09	GHANSHYAM	5	D05	F09	S09
F10	ABHISHEK GV	5	D05	F10	S10
F11	SASI KUMAR	6	D06	F11	S11
F12	L GONDI	6	D06	F12	S12
F13	DON S	7	D07	F13	S13
F14	RAVI	7	D07	F14	S14
F15	GRACE	8	D08	F15	S15
F16	K KAVITA	8	D08	F16	S16
F17	ANKIREDDY	9	D09	F17	S17
F18	MURALI MOHAN	9	D09	F18	S18
F21	JAYA SRI	8	D08	-	-
F23	LUCKY	9	D09	-	-
F19	SUKHIBHAVA	2	D02	-	-
F20	TANGUTOORI	3	D03	-	-
F22	HIMAJA	4	D04	-	-

Download CSV

23 rows selected.

```
-- RIGHT JOIN
SELECT *
FROM TEACHES T
RIGHT JOIN FACULTY F
ON F.FAC_ID = T.FAC_ID;
```

FAC_ID	SUB_ID	FAC_ID	FAC_NAME	ROOM_NO	DEP_ID
F01	S01	F01	PREM SINGH	1	D01
F02	S02	F02	MANSA DEVI	1	D01
F03	S03	F03	BHAVANI	2	D02
F04	S04	F04	SRINIVAS RAO	2	D02
F05	S05	F05	VIJAY SINGH	3	D03
F06	S06	F06	PRAVEEN KUMAR	3	D03
F07	S07	F07	VANDANA SINGH	4	D04
F08	S08	F08	RAGHAVENDRA	4	D04
F09	S09	F09	GHANSHYAM	5	D05
F10	S10	F10	ABHISHEK GV	5	D05
F11	S11	F11	SASI KUMAR	6	D06
F12	S12	F12	L GONDI	6	D06
F13	S13	F13	DON S	7	D07
F14	S14	F14	RAVI	7	D07
F15	S15	F15	GRACE	8	D08
F16	S16	F16	K KAVITA	8	D08
F17	S17	F17	ANKIREDDY	9	D09
F18	S18	F18	MURALI MOHAN	9	D09
-	-	F21	JAYA SRI	8	D08
-	-	F23	LUCKY	9	D09
-	-	F19	SUKHIBHAVA	2	D02
-	-	F20	TANGUTOORI	3	D03
-	-	F22	HIMAJA	4	D04

Download CSV

23 rows selected.

```
-- OUTER JOIN
SELECT *
FROM FACULTY F
FULL OUTER JOIN TEACHES T
ON  F.FAC_ID = T.FAC_ID;
```

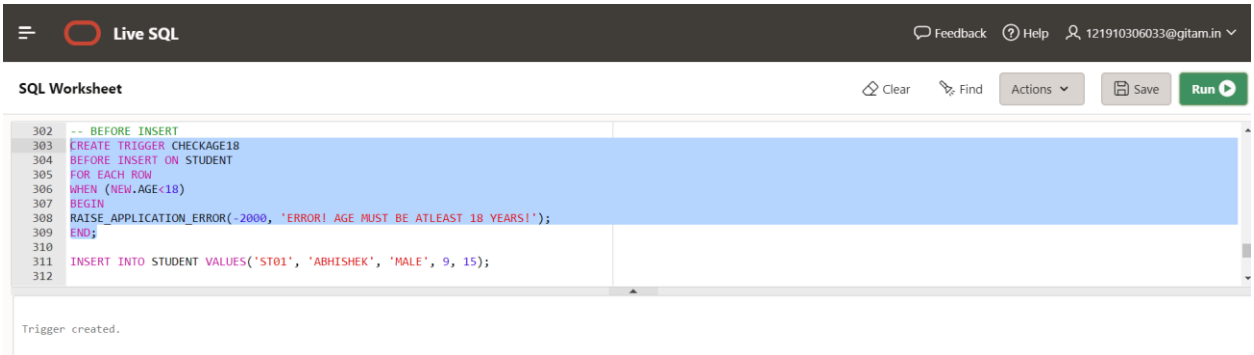
FAC_ID	FAC_NAME	ROOM_NO	DEP_ID	FAC_ID	SUB_ID
F01	PREM SINGH	1	D01	F01	S01
F02	MANSA DEVI	1	D01	F02	S02
F03	BHAVANI	2	D02	F03	S03
F04	SRINIVAS RAO	2	D02	F04	S04
F05	VIJAY SINGH	3	D03	F05	S05
F06	PRAVEEN KUMAR	3	D03	F06	S06
F07	VANDANA SINGH	4	D04	F07	S07
F08	RAGHAVENDRA	4	D04	F08	S08
F09	GHANSHYAM	5	D05	F09	S09
F10	ABHISHEK GV	5	D05	F10	S10
F11	SASI KUMAR	6	D06	F11	S11
F12	L GONDI	6	D06	F12	S12
F13	DON S	7	D07	F13	S13
F14	RAVI	7	D07	F14	S14
F15	GRACE	8	D08	F15	S15
F16	K KAVITA	8	D08	F16	S16
F17	ANKIREDDY	9	D09	F17	S17
F18	MURALI MOHAN	9	D09	F18	S18
F19	SUKHIBHAVA	2	D02	-	-
F20	TANGUTOORI	3	D03	-	-
F21	JAYA SRI	8	D08	-	-
F22	HIMAJA	4	D04	-	-
F23	LUCKY	9	D09	-	-

Download CSV

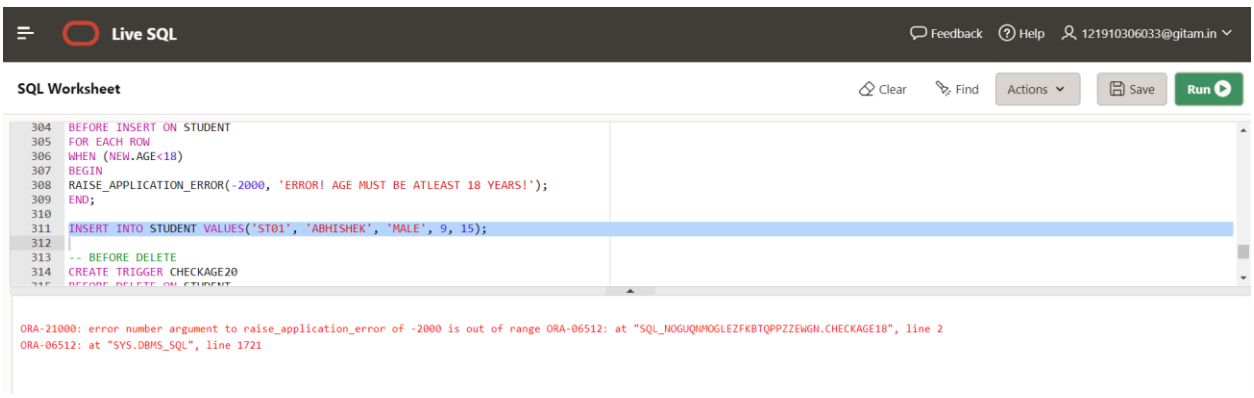
23 rows selected.

Create Triggers For The Database And Cursors

```
-- TRIGGERS
-- BEFORE INSERT
-- CREATE TRIGGER TO CHECK THAT AGE MUST BE ATLEAST 18 YEARS BEFORE INSERTING
CREATE TRIGGER CHECKAGE18
BEFORE INSERT ON STUDENT
FOR EACH ROW
WHEN (NEW.AGE<18)
BEGIN
RAISE_APPLICATION_ERROR(-20000, 'ERROR! AGE MUST BE ATLEAST 18 YEARS!');
END;
```

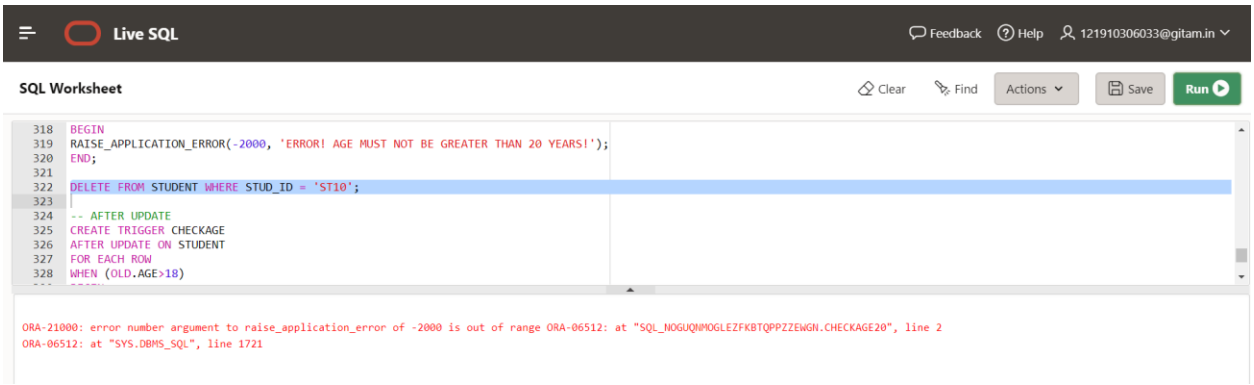



```
INSERT INTO STUDENT VALUES('ST01', 'ABHISHEK', 'MALE', 9, 15);
```



```
-- BEFORE DELETE
-- CREATE TRIGGER TO CHECK THAT AGE MUST NOT BE GREATER THAN 20 YEARS BEFORE DELETING
CREATE TRIGGER CHECKAGE20
BEFORE DELETE ON STUDENT
FOR EACH ROW
WHEN (OLD.AGE>20)
BEGIN
RAISE_APPLICATION_ERROR(-20000, 'ERROR! AGE MUST NOT BE GREATER THAN 20 YEARS!');
END;
```

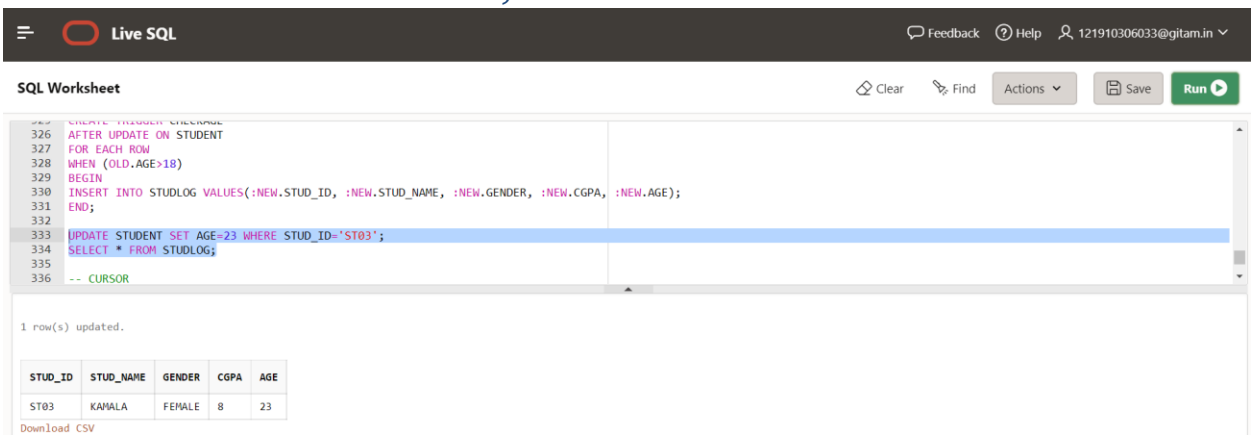

```
DELETE FROM STUDENT WHERE STUD_ID = 'ST10';
```



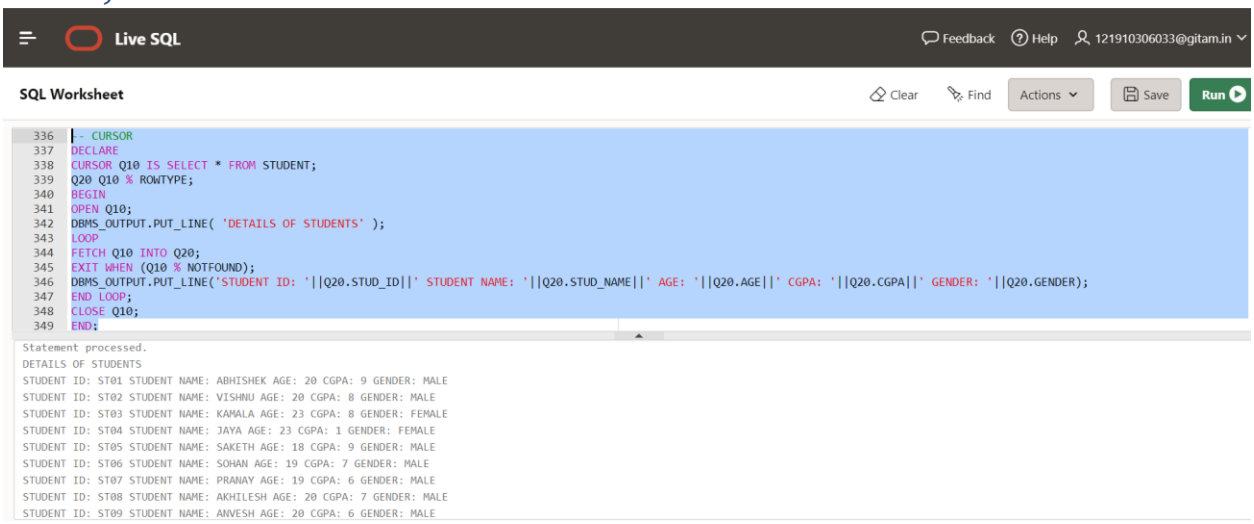
-- AFTER UPDATE
--CREATE TRIGGER TO STORE STUDENT UPDATE LOG AFTER UPDATING IF AGE IS GREATER THAN OR EQUAL TO 18

```
CREATE TRIGGER CHECKAGE
AFTER UPDATE ON STUDENT
FOR EACH ROW
WHEN (OLD.AGE>18)
BEGIN
INSERT INTO STUDLOG VALUES(:NEW.STUD_ID, :NEW.STUD_NAME, :NEW.GENDER, :NEW.CGPA, :NEW.AGE);
END;
```


```
UPDATE STUDENT SET AGE=23 WHERE STUD_ID='ST04';
SELECT * FROM STUD;
```



```
-- CURSOR
-- RETRIEVE DATA FROM STUDENT TABLE AND PRINT IN TEXT FORMAT
DECLARE
CURSOR Q10 IS SELECT * FROM STUDENT;
Q20 Q10 % ROWTYPE;
BEGIN
OPEN Q10;
DBMS_OUTPUT.PUT_LINE( 'DETAILS OF STUDENTS' );
LOOP
FETCH Q10 INTO Q20;
EXIT WHEN (Q10 % NOTFOUND);
DBMS_OUTPUT.PUT_LINE('STUDENT ID: '||Q20.STUD_ID||' STUDENT NAME: '||Q20.STUD_NAME||' AGE: '||Q20.AGE||' CGPA: '||Q20.CGPA||' GENDER: '||Q20.GENDER);
END LOOP;
CLOSE Q10;
END;
```



PART – 5

NORMALIZATION –

For normalization we will consider the tables FACULTY

FAC_ID	FAC_NAME	ROOM_NO	DEP_ID
F01	PREM SINGH	1	D01
F02	MANSA DEVI	1	D01
F03	BHAVANI	2	D02
F04	SRINIVAS RAO	2	D02
F05	VIJAY SINGH	3	D03
F06	PRAVEEN KUMAR	3	D03
F07	VANDANA SINGH	4	D04
F08	RAGHAVENDRA	4	D04
F09	GHANSHYAM	5	D05
F10	ABHISHEK GV	5	D05
F11	SASI KUMAR	6	D06
F12	L GONDI	6	D06
F13	DON S	7	D07
F14	RAVI	7	D07
F15	GRACE	8	D08
F16	K KAVITA	8	D08
F17	ANKIREDDY	9	D09
F18	MURALI MOHAN	9	D09
F19	SUKHIBHAVA	2	D02
F20	TANGUTOORI	3	D03
F21	JAYA SRI	8	D08
F22	HIMAJA	4	D04
F23	LUCKY	9	D09

Download CSV

FACULTY

FACULTY(FAC_ID(PK), FAC_NAME, ROOM_NO, DEP_ID)

FUNCTIONAL DEPENDACIES:

{

FAC_ID → FAC_ID, FAC_NAME

DEP_ID → DEP_ID, ROOM_NO

}

CANDIDATE KEY:

THE CANDIDATE KEY WILL BE: **(FAC_ID, DEP_ID)**

BECAUSE BY PERFORMING **ATTRIBUTE CLOSURE:**

(FAC_ID)$^{+}$ = (FAC_ID,, FAC_NAME)

(DEP_ID)$^{+}$ = (DEP_ID, ROOM_NO)

(FAC_ID, DEP_ID)$^{+}$ = (FAC_ID, FAC_NAME,DEP_ID, ROOM_NO)

ALL ATTRIBUTES ARE DERIVED

THEREFORE CK = **(FAC_ID, DEP_ID)**

PRIME ATTRIBUTES ARE THE ATTRIBUTES PRESENT IN CK-

- **PRIME ATTRIBUTES:** (FAC_ID, DEP_ID)
- **NON-PRIME ATTRIBUTES:** (FAC_NAME, ROOM_NO)

1. 1ST NORMAL FORM

a. All values of each attribute must be ATOMIC
b. No composite values
c. All entries in any column must be of the same kind
d. Each column must have a unique name
e. No two rows are identical

THE FACULTY RELATION IS ALREADY IN 1 NF AS IT SATISFIES THE ABOVE CONDITIONS.

2. 2ND NORMAL FORM

a. The table should be in 1 NF.
b. No partial dependency should be present. Only full dependency should be present.
c. All non-prime attributes should be fully functional dependent on any key (CK) of R, i.e. L.H.S IS A PROPER SUBSET OF CK.

LET US CHECK FOR FACULTY RELATION (R):

FD = {
FAC_ID → FAC_ID, FAC_NAME
DEP_ID → DEP_ID, ROOM_NO
DEP_ID → ROOM_NO
}

IN THE DEPENDACY DEP_ID → ROOM_NO, ROOM_NO IS A NON-PRIME ATTRIBUTE AND IS PARTIALLY DEPENDED ON DEP_ID

THIS VIOLATES THE 2ND NORMAL FORM RULES.

SO, WE HAVE TO **DECOMPOSE** THE RELATION INTO R1 AND R2

R1: (FAC_ID, FAC_NAME, DEP_ID)
FD:
{
FAC_ID → FAC_ID, FAC_NAME
DEP_ID → DEP_ID
}
CK: (FAC_ID, DEP_ID)
SO,
PA: (FAC_ID, DEP_ID)
NPA: (FAC_NAME)

AND

R2: (DEP_ID, ROOM_NO)
FD:
{
DEP_ID → DEP_ID, ROOM_NO
}
CK: (DEP_ID)
SO,
PA: (DEP_ID)
NPA: (ROOM_NO)

NOW BOTH R1 AND R2 SATISFIES 2NF.

NOTE: IT IS A LOSSLESS JOIN DECOMPOSITION AS THE DECOMPOSITION HAS CANDIDATE KEY OF BOTH R1 AND R2 AS ITS COMMON ATTRIBUTE

```
CREATE VIEW ROOMS AS
SELECT DISTINCT DEP_ID, ROOM_NO
FROM FACULTY;

SELECT * FROM ROOMS;

ALTER TABLE FACULTY
DROP COLUMN ROOM_NO;

SELECT * FROM FACULTY;
```

View created.

DEP_ID	ROOM_NO
D03	3
D06	6
D01	1
D04	4
D05	5
D09	9
D07	7
D08	8
D02	2

Download CSV

9 rows selected.

Table altered.

FAC_ID	FAC_NAME	DEP_ID
F01	PREM SINGH	D01
F02	MANSA DEVI	D01
F03	BHAVANI	D02
F04	SRINIVAS RAO	D02
F05	VIJAY SINGH	D03
F06	PRAVEEN KUMAR	D03
F07	VANDANA SINGH	D04
F08	RAGHAVENDRA	D04
F09	GHANSHYAM	D05
F10	ABHISHEK GV	D05
F11	SASI KUMAR	D06
F12	L GONDI	D06
F13	DON S	D07
F14	RAVI	D07

3. 3RD NORMAL FORM

a. The table should be in 2 NF.
b. No transitive dependency should be present. In other words, a prime attribute should not determine a non-prime attribute.
c. No non-prime attribute is depended on another non-prime attribute i.e., L.H.S MUST BE A CK OR SK OR R.H.S IS PA.

SO, 1ST LET'S CHECK 3NF FOR R1 AND R2 -

R1: (FAC_ID, FAC_NAME, DEP_ID)
FD:
{
FAC_ID → FAC_ID, FAC_NAME
DEP_ID → DEP_ID,
}
CK: (FAC_ID, DEP_ID)
PA: (FAC_ID, DEP_ID)
NPA: (FAC_NAME)

AND

R2: (DEP_ID, ROOM_NO)
FD:
{
DEP_ID → DEP_ID, ROOM_NO
}
CK: (DEP_ID)
PA: (DEP_ID)
NPA: (ROOM_NO)

3NF IS SATISFIED FOR R1 AND R2 THEREFORE, 3NF IS THE HIGHEST NORMAL FORM AS WE DID NOT DECOMPOSE

NOTE: IF THERE IS NO CHANCE TO FURTHER DECOMPOSE A TABLE/RELATION, THEN IT IS THE HIGHEST NORMAL FORM OF THAT TABLE.
SO, WE NEED NOT CONTINUE TO CHECK R2 AND R1 FOR FURTHER NORMALIZATION.

4. 3.5 NORMAL FORM OR BCNF

a. The table should be in 3NF.
b. Every attribute must be dependent on a super key, i.e. all LHS values must be either candidate or super key.

SO, 1ST LET'S CHECK 3NF FOR R1 AND R2 -

R1: (FAC_ID, FAC_NAME, DEP_ID)
FD:
{
FAC_ID → FAC_ID, FAC_NAME
DEP_ID → DEP_ID,
}
CK: (FAC_ID, DEP_ID)
PA: (FAC_ID, DEP_ID)
NPA: (FAC_NAME)

AND

R2: (DEP_ID, ROOM_NO)
FD:
{
DEP_ID → DEP_ID, ROOM_NO
}
CK: (DEP_ID)
PA: (DEP_ID)
NPA: (ROOM_NO)

3.5NF IS SATISFIED FOR R1 AND R2

5. 4TH NORMAL FORM

a. The table should be in BCNF.
b. No multivalued dependency should be present (x->->y)
c. For Table with ABC columns, B and C should be independent

4NF IS SATISFIED FOR R1 AND R2

6. 5TH NORMAL FORM

a. The table should be in 4NF.
b. There should only be lossless join dependency.

5NF IS SATISFIED FOR R1 AND R2

PART – 5 JDBC
JAVA DATABASE CONNECTIVITY
Before getting to the code

SQL Plus

```
SQL> create table student33(sid integer primary key,sname varchar(20),rno integer);

Table created.

SQL> insert into student33 values(1,'prateek',33);

1 row created.

SQL> insert into student33 values(2,'dhorababu',21);

1 row created.

SQL> insert into student33 values(3,'surya',37);

1 row created.

SQL> insert into student33 values(4,'gowtham',52);

1 row created.

SQL> select * from student33;

       SID SNAME                       RNO
---------- -------------------- ----------
         1 prateek                      33
         2 dhorababu                    21
         3 surya                        37
         4 gowtham                      52
```

SQL Plus

```
SQL> insert into student33 values(1,'prateek',33);

1 row created.

SQL> insert into student33 values(2,'dhorababu',21);

1 row created.

SQL> insert into student33 values(3,'surya',37);

1 row created.

SQL> insert into student33 values(4,'gowtham',52);

1 row created.

SQL> select * from student33;

       SID SNAME                       RNO
---------- -------------------- ----------
         1 prateek                      33
         2 dhorababu                    21
         3 surya                        37
         4 gowtham                      52

SQL> commit;

Commit complete.

SQL>
```

The programming involved to establish a JDBC connection is fairly simple. Here are these simple four steps −

- Import JDBC Packages − Add import statements to your Java program to import required classes in your Java code.
- Register JDBC Driver − This step causes the JVM to load the desired driver implementation into memory so it can fulfil your JDBC requests.
- Database URL Formulation − This is to create a properly formatted address that points to the database to which you wish to connect.
- Create Connection Object − Finally, code a call to the *DriverManager* object's *getConnection( )* method to establish actual database connection.

Import JDBC packages

The Import statements tell the Java compiler where to find the classes you reference in your code and are placed at the very beginning of your source code.

Import java.sql.*;//for standard JDBC programs
Import java.sql.Connection;
Import java.sql.DriverManager;

Register JDBC Driver

Registering the driver is the process by which the database driver's class file is loaded into the memory, so it can be utilized as an implementation of the JDBC interfaces.

Database URL Formulation

After you've loaded the driver, you can establish a connection using the DriverManager.getConnection() method. For easy reference, let me list the three overloaded DriverManager.getConnection() methods

- getConnection(String url)
- getConnection(String url, Properties prop)
- getConnection(String url, String user, String password)

Create Connection Object

We have listed down three forms
of DriverManager.getConnection() method to create a connection object.

Using a Database URL with a username and password

The most commonly used form of getConnection() requires you to pass a database URL, a *username*, and a *password* −

Assuming you are using Oracle's thin driver, you'll specify a host:port:databaseName value for the database portion of the URL.

Now you have to call getConnection() method with appropriate username and password to get a Connection object as follows –

```
String url=”jdbc:oracle://localhost:5521/orcl”;
String username=”system”;
String password=”system”;
```

Using Only a Database URL

A second form of the DriverManager.getConnection( ) method requires only a database URL −

```
DriverManager.getConnection(String url);
```

Closing JDBC Connections

At the end of your JDBC program, it is required explicitly to close all the connections to the database to end each database session. However, if you forget, Java's garbage collector will close the connection when it cleans up stale objects.

Relying on the garbage collection, especially in database programming, is a very poor programming practice. You should make a habit of always closing the connection with the close() method associated with connection object.

To ensure that a connection is closed, you could provide a 'finally' block in your code. A *finally* block always executes, regardless of an exception occurs or not.

To close the above opened connection, you should call close() method as follows −

```
con.close();
```

Creating Statement Object

Before you can use a Statement object to execute a SQL statement, you need to create one using the Connection object's createStatement( ) method.in the following example

```
Statement st= null;
try {
   st = conn.createStatement( );
   . . .
}
catch (Exception e) {
   . . .
}
finally {
   . . .
}
```

- int executeUpdate (String SQL) − Returns the number of rows affected by the execution of the SQL statement. Use this method to execute SQL statements for which you expect to get a number of rows affected - for example, an INSERT, UPDATE, or DELETE statement.
- ResultSet executeQuery (String SQL) − Returns a ResultSet object. Use this method when you expect to get a result set, as you would with a SELECT statement.

Closing Statement Object

A simple call to the close() method will do the job. If you close the Connection object first, it will close the Statement object as well. However, you should always explicitly close the Statement object to ensure proper cleanup.

St.close()

JAVA CODE

```
package jdbc;

import java.sql.*;

class OracleCon {
   public static void main(String args[]) {
          try {
//step1 load the driver class
                Class.forName("oracle.jdbc.driver.OracleDriver");

//step2 create  the connection object
                Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system",
"oracle");

//step3 create the statement object
                Statement stmt = con.createStatement();

//step4 execute query
                ResultSet rs = stmt.executeQuery("select * from student");
                while (rs.next())
                      System.out.println(rs.getInt(1) + "  " + rs.getString(2) +
"  " + rs.getString(3) + " "
                                        + rs.getString(4) + " " + rs.getString(5));


//step5 close the connection object
                con.close();

          } catch (Exception e) {
                System.out.println(e);
          }

   }
}
```

OUTPUT:

```
<terminated> UniversityJDBC [Java Application]
ST10 ABHISHEK MALE 9 20
ST10 VISHNU MALE 8 20
ST10 KAMALA FEMALE 8 19
ST10 JAYA FEMALE 1 18
ST10 SAKETH MALE 9 18
ST10 SOHAN MALE 7 19
ST10 PRANAY MALE 6 19
ST10 AKHILESH MALE 7 20
ST10 ANVESH MALE 6 20
ST10 PRATHEK MALE 9 21
```

COMPLETE CODE

```
CREATE TABLE UNIVERSITY(BRANCH_ID VARCHAR(5) PRIMARY KEY,
BRANCH_NAME VARCHAR(20) UNIQUE NOT NULL, BRANCH_ADDRESS
VARCHAR(20));

CREATE TABLE DEPARTMENT(DEP_ID VARCHAR(5) PRIMARY KEY, DEP_NAME
VARCHAR(20)  NOT NULL);

CREATE TABLE FACULTY(FAC_ID VARCHAR(5) PRIMARY KEY, FAC_NAME
VARCHAR(20) NOT NULL, ROOM_NO INTEGER, DEP_ID VARCHAR(5), FOREIGN
KEY (DEP_ID) REFERENCES DEPARTMENT);

CREATE TABLE SUBJECT(SUB_ID VARCHAR(5) PRIMARY KEY, SUB_NAME
VARCHAR(20), FAC_ID VARCHAR(5), FOREIGN KEY (FAC_ID) REFERENCES
FACULTY,
DEP_ID VARCHAR(5), FOREIGN KEY (DEP_ID) REFERENCES DEPARTMENT);

CREATE TABLE TEACHES(FAC_ID VARCHAR(5), SUB_ID VARCHAR(5), FOREIGN
KEY (FAC_ID) REFERENCES FACULTY, FOREIGN KEY (SUB_ID) REFERENCES
SUBJECT);

CREATE TABLE STUDENT(STUD_ID VARCHAR(5) PRIMARY KEY, STUD_NAME
VARCHAR(20) NOT NULL, GENDER VARCHAR(10), CGPA INTEGER, AGE INTEGER
CHECK (AGE>=18));

CREATE TABLE STUDLOG(STUD_ID VARCHAR(5) PRIMARY KEY, STUD_NAME
VARCHAR(20) NOT NULL, GENDER VARCHAR(10), CGPA INTEGER, AGE INTEGER
CHECK (AGE>=18));

CREATE TABLE ENROLLED(STUD_ID VARCHAR(5), DEP_ID VARCHAR(5), PRIMARY
KEY(STUD_ID, DEP_ID), FOREIGN KEY (DEP_ID) REFERENCES DEPARTMENT,
FOREIGN KEY (STUD_ID) REFERENCES STUDENT);

CREATE TABLE EXAM(EXAM_ID VARCHAR(5) PRIMARY KEY, SUB_ID
VARCHAR(5), FOREIGN KEY (SUB_ID) REFERENCES SUBJECT);

CREATE TABLE GRADES(STUD_ID VARCHAR(5), EXAM_ID VARCHAR(5), PRIMARY
KEY(STUD_ID, EXAM_ID), FOREIGN KEY (EXAM_ID) REFERENCES EXAM,
```

```
FOREIGN KEY (STUD_ID) REFERENCES STUDENT, MARKS INTEGER, GRADE
VARCHAR(2));

CREATE TABLE ACCOUNTS(STUD_ID VARCHAR(5), FEE INTEGER, STATUS
VARCHAR(3), FOREIGN KEY (STUD_ID) REFERENCES STUDENT);

INSERT INTO UNIVERSITY VALUES('B01', 'GITAM, VSKP', 'VISAKHAPATNAM');
INSERT INTO UNIVERSITY VALUES('B02', 'GITAM, BANGLORE', 'BANGLORE');
INSERT INTO UNIVERSITY VALUES('B03', 'GITAM, HYD', 'HYDERABAD');

SELECT * FROM UNIVERSITY;

INSERT INTO DEPARTMENT VALUES('D01', 'CSE');
INSERT INTO DEPARTMENT VALUES('D02', 'MECH');
INSERT INTO DEPARTMENT VALUES('D03', 'DENTAL');
INSERT INTO DEPARTMENT VALUES('D04', 'LAW');
INSERT INTO DEPARTMENT VALUES('D05', 'ECE');
INSERT INTO DEPARTMENT VALUES('D06', 'PHARMACY');
INSERT INTO DEPARTMENT VALUES('D07', 'EEE');
INSERT INTO DEPARTMENT VALUES('D08', 'MEDICAL');
INSERT INTO DEPARTMENT VALUES('D09', 'ARCH');

SELECT * FROM DEPARTMENT;

INSERT INTO FACULTY VALUES('F01', 'PREM SINGH', '1', 'D01');
INSERT INTO FACULTY VALUES('F02', 'MANSA DEVI', '1', 'D01');
INSERT INTO FACULTY VALUES('F03', 'BHAVANI', '2', 'D02');
INSERT INTO FACULTY VALUES('F04', 'SRINIVAS RAO', '2', 'D02');
INSERT INTO FACULTY VALUES('F05', 'VIJAY SINGH', '3', 'D03');
INSERT INTO FACULTY VALUES('F06', 'PRAVEEN KUMAR', '3', 'D03');
INSERT INTO FACULTY VALUES('F07', 'VANDANA SINGH', '4', 'D04');
INSERT INTO FACULTY VALUES('F08', 'RAGHAVENDRA', '4', 'D04');
INSERT INTO FACULTY VALUES('F09', 'GHANSHYAM', '5', 'D05');
INSERT INTO FACULTY VALUES('F10', 'ABHISHEK GV', '5', 'D05');
INSERT INTO FACULTY VALUES('F11', 'SASI KUMAR', '6', 'D06');
INSERT INTO FACULTY VALUES('F12', 'L GONDI', '6', 'D06');
INSERT INTO FACULTY VALUES('F13', 'DON S', '7', 'D07');
INSERT INTO FACULTY VALUES('F14', 'RAVI', '7', 'D07');
INSERT INTO FACULTY VALUES('F15', 'GRACE', '8', 'D08');
INSERT INTO FACULTY VALUES('F16', 'K KAVITA', '8', 'D08');
```

```
INSERT INTO FACULTY VALUES('F17', 'ANKIREDDY', '9', 'D09');
INSERT INTO FACULTY VALUES('F18', 'MURALI MOHAN', '9', 'D09');
INSERT INTO FACULTY VALUES('F19', 'SUKHIBHAVA', '2', 'D02');
INSERT INTO FACULTY VALUES('F20', 'TANGUTOORI', '3', 'D03');
INSERT INTO FACULTY VALUES('F21', 'JAYA SRI', '8', 'D08');
INSERT INTO FACULTY VALUES('F22', 'HIMAJA', '4', 'D04');
INSERT INTO FACULTY VALUES('F23', 'LUCKY', '9', 'D09');

SELECT * FROM FACULTY;

INSERT INTO SUBJECT VALUES('S01', 'ADV DATA STRUCTURES', 'F01', 'D01');
INSERT INTO SUBJECT VALUES('S02', 'DBMS', 'F02', 'D01');
INSERT INTO SUBJECT VALUES('S03', 'THERMO DYNAMICS', 'F03', 'D02');
INSERT INTO SUBJECT VALUES('S04', 'PHYSICS', 'F04', 'D02');
INSERT INTO SUBJECT VALUES('S05', 'TEETH', 'F05', 'D03');
INSERT INTO SUBJECT VALUES('S06', 'GUMS', 'F06', 'D03');
INSERT INTO SUBJECT VALUES('S07', 'CIVICS', 'F07', 'D04');
INSERT INTO SUBJECT VALUES('S08', 'CRIMINOLOGY', 'F08', 'D04');
INSERT INTO SUBJECT VALUES('S09', 'FDLC', 'F09', 'D05');
INSERT INTO SUBJECT VALUES('S10', 'COA', 'F10', 'D05');
INSERT INTO SUBJECT VALUES('S11', 'LIFE SCIENCES', 'F11', 'D06');
INSERT INTO SUBJECT VALUES('S12', 'BIO CHEMISTRY', 'F11', 'D06');
INSERT INTO SUBJECT VALUES('S13', 'BEEE', 'F13', 'D07');
INSERT INTO SUBJECT VALUES('S14', 'ELECTRICAL', 'F14', 'D07');
INSERT INTO SUBJECT VALUES('S15', 'HUMAN BODY', 'F15', 'D08');
INSERT INTO SUBJECT VALUES('S16', 'REPRODUCTION', 'F16', 'D08');
INSERT INTO SUBJECT VALUES('S17', 'SCALES', 'F17', 'D09');
INSERT INTO SUBJECT VALUES('S18', 'AUTOCAD', 'F18', 'D09');

SELECT * FROM SUBJECT;

INSERT INTO STUDENT VALUES('ST01', 'ABHISHEK', 'MALE', 9, 20);
INSERT INTO STUDENT VALUES('ST02', 'VISHNU', 'MALE', 8, 20);
INSERT INTO STUDENT VALUES('ST03', 'KAMALA', 'FEMALE', 8, 19);
INSERT INTO STUDENT VALUES('ST04', 'JAYA', 'FEMALE', 1, 18);
INSERT INTO STUDENT VALUES('ST05', 'SAKETH', 'MALE', 9, 18);
INSERT INTO STUDENT VALUES('ST06', 'SOHAN', 'MALE', 7, 19);
INSERT INTO STUDENT VALUES('ST07', 'PRANAY', 'MALE', 6, 19);
INSERT INTO STUDENT VALUES('ST08', 'AKHILESH', 'MALE', 7, 20);
INSERT INTO STUDENT VALUES('ST09', 'ANVESH', 'MALE', 6, 20);
```

```
INSERT INTO STUDENT VALUES('ST10', 'PRATHEK', 'MALE', 10, 21);

SELECT * FROM STUDENT;

INSERT INTO TEACHES VALUES('F01', 'S01');
INSERT INTO TEACHES VALUES('F02', 'S02');
INSERT INTO TEACHES VALUES('F03', 'S03');
INSERT INTO TEACHES VALUES('F04', 'S04');
INSERT INTO TEACHES VALUES('F05', 'S05');
INSERT INTO TEACHES VALUES('F06', 'S06');
INSERT INTO TEACHES VALUES('F07', 'S07');
INSERT INTO TEACHES VALUES('F08', 'S08');
INSERT INTO TEACHES VALUES('F09', 'S09');
INSERT INTO TEACHES VALUES('F10', 'S10');
INSERT INTO TEACHES VALUES('F11', 'S11');
INSERT INTO TEACHES VALUES('F12', 'S12');
INSERT INTO TEACHES VALUES('F13', 'S13');
INSERT INTO TEACHES VALUES('F14', 'S14');
INSERT INTO TEACHES VALUES('F15', 'S15');
INSERT INTO TEACHES VALUES('F16', 'S16');
INSERT INTO TEACHES VALUES('F17', 'S17');
INSERT INTO TEACHES VALUES('F18', 'S18');

SELECT* FROM TEACHES;

INSERT INTO ENROLLED VALUES('ST01', 'D01');
INSERT INTO ENROLLED VALUES('ST02', 'D02');
INSERT INTO ENROLLED VALUES('ST03', 'D03');
INSERT INTO ENROLLED VALUES('ST04', 'D04');
INSERT INTO ENROLLED VALUES('ST05', 'D05');
INSERT INTO ENROLLED VALUES('ST06', 'D06');
INSERT INTO ENROLLED VALUES('ST07', 'D07');
INSERT INTO ENROLLED VALUES('ST08', 'D08');
INSERT INTO ENROLLED VALUES('ST09', 'D09');
INSERT INTO ENROLLED VALUES('ST10', 'D01');
INSERT INTO ENROLLED VALUES('ST04', 'D01');
INSERT INTO ENROLLED VALUES('ST05', 'D07');
INSERT INTO ENROLLED VALUES('ST06', 'D08');

SELECT* FROM ENROLLED;
```

```
INSERT INTO EXAM VALUES('E01', 'S01');
INSERT INTO EXAM VALUES('E02', 'S02');
INSERT INTO EXAM VALUES('E03', 'S03');
INSERT INTO EXAM VALUES('E04', 'S04');
INSERT INTO EXAM VALUES('E05', 'S05');
INSERT INTO EXAM VALUES('E06', 'S06');
INSERT INTO EXAM VALUES('E07', 'S07');
INSERT INTO EXAM VALUES('E08', 'S08');
INSERT INTO EXAM VALUES('E09', 'S09');
INSERT INTO EXAM VALUES('E10', 'S10');
INSERT INTO EXAM VALUES('E11', 'S11');
INSERT INTO EXAM VALUES('E12', 'S12');
INSERT INTO EXAM VALUES('E13', 'S13');
INSERT INTO EXAM VALUES('E14', 'S14');
INSERT INTO EXAM VALUES('E15', 'S15');
INSERT INTO EXAM VALUES('E16', 'S16');
INSERT INTO EXAM VALUES('E17', 'S17');
INSERT INTO EXAM VALUES('E18', 'S18');

SELECT * FROM EXAM;

INSERT INTO GRADES VALUES('ST01', 'E01' ,95, 'A');
INSERT INTO GRADES VALUES('ST01', 'E02' ,85, 'B');
INSERT INTO GRADES VALUES('ST02', 'E03' ,95, 'A');
INSERT INTO GRADES VALUES('ST02', 'E04' ,85, 'B');
INSERT INTO GRADES VALUES('ST03', 'E05' ,75, 'C');
INSERT INTO GRADES VALUES('ST03', 'E06' ,75, 'C');
INSERT INTO GRADES VALUES('ST04', 'E07' ,75, 'C');
INSERT INTO GRADES VALUES('ST04', 'E08' ,95, 'A');
INSERT INTO GRADES VALUES('ST05', 'E09' ,85, 'B');
INSERT INTO GRADES VALUES('ST05', 'E10' ,85, 'B');
INSERT INTO GRADES VALUES('ST06', 'E11' ,95, 'A');
INSERT INTO GRADES VALUES('ST06', 'E12' ,65, 'D');
INSERT INTO GRADES VALUES('ST07', 'E13' ,65, 'D');
INSERT INTO GRADES VALUES('ST07', 'E14' ,95, 'A');
INSERT INTO GRADES VALUES('ST08', 'E15' ,75, 'C');
INSERT INTO GRADES VALUES('ST08', 'E16' ,85, 'B');
INSERT INTO GRADES VALUES('ST09', 'E17' ,85, 'B');
INSERT INTO GRADES VALUES('ST10', 'E18' ,95, 'A');
```

```
SELECT * FROM GRADES;

INSERT INTO ACCOUNTS VALUES('ST01', 270000, 'YES');
INSERT INTO ACCOUNTS VALUES('ST02', 250000, 'YES');
INSERT INTO ACCOUNTS VALUES('ST03', 250000, 'NO');
INSERT INTO ACCOUNTS VALUES('ST04', 210000, 'NO');
INSERT INTO ACCOUNTS VALUES('ST05', 170000, 'YES');
INSERT INTO ACCOUNTS VALUES('ST06', 350000, 'YES');
INSERT INTO ACCOUNTS VALUES('ST07', 320000, 'NO');
INSERT INTO ACCOUNTS VALUES('ST08', 270000, 'YES');
INSERT INTO ACCOUNTS VALUES('ST09', 270000, 'YES');
INSERT INTO ACCOUNTS VALUES('ST10', 220000, 'NO');
SELECT * FROM ACCOUNTS;

ALTER TABLE ACCOUNTS ADD SCHOLARSHIP INTEGER;
SELECT * FROM ACCOUNTS;

ALTER TABLE ACCOUNTS DROP COLUMN SCHOLARSHIP;
SELECT * FROM ACCOUNTS;

ALTER TABLE ACCOUNTS MODIFY STATUS VARCHAR(3) NOT NULL;
DESC ACCOUNTS;

UPDATE ACCOUNTS SET STATUS='YES' WHERE STUD_ID='ST10';
SELECT * FROM ACCOUNTS;

DELETE FROM ACCOUNTS WHERE STUD_ID='ST09';
SELECT * FROM ACCOUNTS;

SELECT MIN(FEE) FROM ACCOUNTS;
SELECT MAX(FEE) FROM ACCOUNTS;
SELECT AVG(FEE) FROM ACCOUNTS;
SELECT COUNT(FEE) FROM ACCOUNTS;
SELECT SUM(FEE) FROM ACCOUNTS;

-- PRINT NAMES OF FACULTY WHO BELONG TO CSE DEPARTMENT
SELECT FAC_NAME FROM FACULTY WHERE DEP_ID IN (SELECT DEP_ID FROM
DEPARTMENT WHERE DEP_NAME='CSE');

-- PRINT NAMES OF FACULTY WHO DO NOT BELONG TO CSE DEPARTMENT
```

```
SELECT FAC_NAME FROM FACULTY WHERE DEP_ID NOT IN (SELECT DEP_ID
FROM DEPARTMENT WHERE DEP_NAME='CSE');

-- PRINT NAMES OF STUDENTS HAVING 'A' GRADE
SELECT S.STUD_NAME FROM STUDENT S WHERE EXISTS (SELECT G.STUD_ID
FROM GRADES G WHERE S.STUD_ID = G.STUD_ID AND G.GRADE='A');

-- PRINT NAMES OF STUDENTS NOT HAVING 'A' GRADE
SELECT S.STUD_NAME FROM STUDENT S WHERE NOT EXISTS (SELECT G.STUD_ID
FROM GRADES G WHERE S.STUD_ID = G.STUD_ID AND G.GRADE='A');

-- PRINT NAMES OF STUDENTS WHO SCORED MORE THAN 85 MARKS
SELECT DISTINCT S.STUD_NAME FROM STUDENT S, GRADES G WHERE S.STUD_ID
= G.STUD_ID AND G.MARKS>85;

-- PRINT NAMES OF STUDENTS WHO SCORED LESS THAN 85 MARKS
SELECT DISTINCT S.STUD_NAME FROM STUDENT S, GRADES G WHERE S.STUD_ID
= G.STUD_ID AND G.MARKS<85;

-- PRINT NAMES OF STUDENTS WHO DID NOT SCORE 85 MARKS
SELECT DISTINCT S.STUD_NAME FROM STUDENT S, GRADES G WHERE S.STUD_ID
= G.STUD_ID AND G.MARKS<>85;

-- PRINT NAMES OF STUDENTS WHO SCORED MORE MARKS GREATER THAN
EQUAL TO ALL OTHER STUDENTS
SELECT DISTINCT S.STUD_NAME, G.MARKS FROM STUDENT S, GRADES G WHERE
S.STUD_ID = G.STUD_ID AND G.MARKS>=ALL(SELECT MARKS FROM GRADES);

-- PRINT NAMES OF STUDENTS WHO SCORED MORE MARKS GREATER THAN ANY
ONE OTHER STUDENT
SELECT DISTINCT S.STUD_NAME, G.MARKS FROM STUDENT S, GRADES G WHERE
S.STUD_ID = G.STUD_ID AND G.MARKS>ANY(SELECT MARKS FROM GRADES);

--PRINT NAMES OF STUDENTS WHO ENROLLED IN EITHER 'CSE' OR 'EEE'
SELECT S.STUD_NAME FROM STUDENT S, ENROLLED E WHERE S.STUD_ID =
E.STUD_ID AND E.DEP_ID = 'D01' UNION
SELECT S.STUD_NAME FROM STUDENT S, ENROLLED E WHERE S.STUD_ID =
E.STUD_ID AND E.DEP_ID = 'D07' ;

--PRINT NAMES OF STUDENTS WHO ENROLLED IN BOTH 'CSE' AND 'LAW'
```

```
SELECT S.STUD_NAME FROM STUDENT S, ENROLLED E WHERE S.STUD_ID =
E.STUD_ID AND E.DEP_ID = 'D01' INTERSECT
SELECT S.STUD_NAME FROM STUDENT S, ENROLLED E WHERE S.STUD_ID =
E.STUD_ID AND E.DEP_ID = 'D04' ;

--PRINT NAMES OF STUDENTS WHO ENROLLED IN ONLY 'CSE' NOT 'LAW'
SELECT S.STUD_NAME FROM STUDENT S, ENROLLED E WHERE S.STUD_ID =
E.STUD_ID AND E.DEP_ID = 'D01' MINUS
SELECT S.STUD_NAME FROM STUDENT S, ENROLLED E WHERE S.STUD_ID =
E.STUD_ID AND E.DEP_ID = 'D04' ;

--COUNT GROUPS OF STUDENTS BASED ON STATUS OF FEE PAYMENT
SELECT STATUS, COUNT(STATUS) FROM ACCOUNTS GROUP BY STATUS;

--COUNT GROUPS OF STUDENTS BASED ON STATUS OF FEE PAYMENT IN
ASCENDING ORDER OF COUNT
SELECT STATUS, COUNT(STATUS) FROM ACCOUNTS GROUP BY STATUS ORDER
BY COUNT(STATUS);

--COUNT GROUPS OF STUDENTS BASED ON STATUS OF FEE PAYMENT WHO HAVE
NOT PAID THE FEE
SELECT STATUS, COUNT(STATUS) FROM ACCOUNTS GROUP BY STATUS HAVING
STATUS = 'NO';

--PRINT NAMES OF STUDENTS HAVING AGE IN BETWEEN 21 AND 21
SELECT STUD_NAME, AGE FROM STUDENT WHERE AGE BETWEEN 20 AND 21;

-- NATURAL JOIN
SELECT *
FROM FACULTY F, TEACHES T
WHERE F.FAC_ID = T.FAC_ID;

-- INNER JOIN
SELECT *
FROM FACULTY F
INNER JOIN TEACHES T
ON F.FAC_ID = T.FAC_ID;

-- LEFT JOIN
SELECT *
```

```
FROM FACULTY F
LEFT JOIN TEACHES T
ON F.FAC_ID = T.FAC_ID;

-- RIGHT JOIN
SELECT *
FROM TEACHES T
RIGHT JOIN FACULTY F
ON F.FAC_ID = T.FAC_ID;

-- OUTER JOIN
SELECT *
FROM FACULTY F
FULL OUTER JOIN TEACHES T
ON  F.FAC_ID = T.FAC_ID;

-- TRIGGERS
-- BEFORE INSERT
CREATE TRIGGER CHECKAGE18
BEFORE INSERT ON STUDENT
FOR EACH ROW
WHEN (NEW.AGE<18)
BEGIN
RAISE_APPLICATION_ERROR(-2000, 'ERROR! AGE MUST BE ATLEAST 18 YEARS!');
END;

INSERT INTO STUDENT VALUES('ST01', 'ABHISHEK', 'MALE', 9, 15);

-- BEFORE DELETE
CREATE TRIGGER CHECKAGE20
BEFORE DELETE ON STUDENT
FOR EACH ROW
WHEN (OLD.AGE>20)
BEGIN
RAISE_APPLICATION_ERROR(-2000, 'ERROR! AGE MUST NOT BE GREATER THAN
20 YEARS!');
END;

DELETE FROM STUDENT WHERE STUD_ID = 'ST10';
```

```
-- AFTER UPDATE
CREATE TRIGGER CHECKAGE
AFTER UPDATE ON STUDENT
FOR EACH ROW
WHEN (OLD.AGE>18)
BEGIN
INSERT INTO STUDLOG VALUES(:NEW.STUD_ID, :NEW.STUD_NAME,
:NEW.GENDER, :NEW.CGPA, :NEW.AGE);
END;

UPDATE STUDENT SET AGE=23 WHERE STUD_ID='ST04';
SELECT * FROM STUD;

-- CURSOR
DECLARE
CURSOR Q10 IS SELECT * FROM STUDENT;
Q20 Q10 % ROWTYPE;
BEGIN
OPEN Q10;
DBMS_OUTPUT.PUT_LINE( 'DETAILS OF STUDENTS' );
LOOP
FETCH Q10 INTO Q20;
EXIT WHEN (Q10 % NOTFOUND);
DBMS_OUTPUT.PUT_LINE('STUDENT ID: '||Q20.STUD_ID||' STUDENT NAME:
'||Q20.STUD_NAME||' AGE: '||Q20.AGE||' CGPA: '||Q20.CGPA||' GENDER: '||Q20.GENDER);
END LOOP;
CLOSE Q10;
END;

CREATE VIEW ROOMS AS
SELECT DISTINCT DEP_ID, ROOM_NO
FROM FACULTY;

ALTER TABLE FACULTY
DROP COLUMN ROOM_NO;

SELECT * FROM ROOMS;
SELECT * FROM FACULTY;
```