

CS 432: Databases

Assignment 2: DEVELOPING THE DBMS

Group Name: La Retro's

Topic: Placement Management

Group Members-

G1: Aaryan Darad - 21110001
Abhay Kumar Upparwal - 21110004
Aditya Deshmukh - 21110014
Anugu Arun Reddy - 21110029

G2: Adit Kaushik - 21110010
Gaurav Joshi - 21110065
Ahaan Giriya - 21110015
Rutwik More - 21110133

Important Links

- 1) ER Diagram: Draw.io file:

<https://app.diagrams.net/?client=1#G1yCpFEy0jyzqNLsKFg6Lim8za2PPASyPd%7B%22pageId%22%3A%22n4RG28VcgylvL5a97sLb%22%7D>

- 2) Code for the Data Generation process: Google Colab notebook:

<https://colab.research.google.com/drive/1thD8FtA0pBjFwbtnWxz85L4WhKXOa4s->

- 3) Generated Dataset for the Database: Google Drive:

https://drive.google.com/drive/folders/1lr40jKZI_w58t7_DY7AVztuXKtXFc2A2

- 4) SQL Files, and data dump:

https://drive.google.com/drive/folders/1_MqzNJfXfNdrbH-PaVhR7fRd7r1Yt9vL

3.1 Responsibility of G1:

- 1. Populate the tables that you created in the previous assignment with random data with the following constraints. All tables must follow the ACID properties and the previous constraints mentioned in Assignment 1.*

Populating the tables with data was a pivotal aspect of our project. To generate diverse data for various entities, we utilized the Faker and Random libraries in Python. These libraries enabled us to create synthetic data, which we then organized into CSV files. Subsequently, we imported these CSV files into MySQL Workbench to populate the corresponding tables.

This process presented its challenges, as we had to meticulously handle various constraints. Ensuring the uniqueness of primary keys across tables was imperative, alongside managing different types of relationships such as total/partial, one-one, many-one, and one-many participation. Additionally, we paid close attention to using appropriate data types for each table.

To streamline this task, we devised individual scripts for each table. Initially, we focused on populating the entity tables. Once completed, we leveraged these entity tables to populate the relationship tables while adhering to their specific constraints. Finally, we imported all the CSV files containing the populated tables into MySQL.

- How do we ensure the primary key is not repeated?

Let's say we are generating company entity data:

I have used the below code where we keep track of the ids/company_ids which is the primary key for the company table. If the generated id is company_ids_ we generate it again till it is not in company_ids_

- How do we ensure constraints are being followed?

In the below code, if we see for generating, we have used faker, and we generate the name of the company till it follows the mentioned constraint i.e. len(company_name)<255.

```

company_ids_ = []
def generate_company_data():
    company_id = random.randint(0, 9999)
    while company_id in company_ids_:
        company_id = random.randint(0, 9999)
    company_ids_.append(company_id)
    company_name = fake.company()
    if len(company_name)>=255:
        while(len(company_name)>=255):
            company_name = fake.company()
    company_website = company_name.replace(' ', '').lower() + '.com'
    organization_type = random.choice(["Public", "Private", "Non-Profit"])
    company_field = random.choice(["Tech", "Data", "Finance", "Design"])
    company_brochure = company_name.replace(' ', '').lower() + '_brochure.pdf'
    country_of_origin = random.choice(["USA", "Canada", "UK", "Australia", "Germany", "France", "Japan", "India", "China"])
    return [company_id, company_name, company_website, organization_type, company_field, company_brochure, country_of_origin]

company_entries = []
for _ in range(50):
    cur = generate_company_data()
    if cur in company_entries:
        while cur in company_entries:
            cur = generate_company_data()
    company_entries.append(cur)

```

Similarly, we generated data for all the entities.

Once that is done, we synthesize data for the relationships.

→ How do we ensure cardinality is being followed?

Let us take the example below. In this, the relationship between company and opportunity is one to many while opportunity is having total participation.

To ensure the above condition of total participation, we iterated through every available opportunity and assigned randomly to any company

```

opportunity_company_entries = []
for j in range(len(opportunity_entries)):
    comp_id = random.sample(company_entries, 1)[0]
    opportunity_company_entries.append([comp[0], opportunity_entries[j][0]])
    comp_opp_dict[comp[0]].append(opportunity_entries[j][0])

```

→ In the below case to ensure the relationship between the company and POC is one to many with total participation on both sides.

To do the above we iterated through every POC and assigned POC to every company till every company is assigned POC and the remaining POC is randomly assigned.

```

poc_company_entries = []
for j in range(len(person_of_contact_entries)):
    if j<len(company_entries):
        poc_company_entries.append([company_entries[j][0],person_of_contact_entries[j][0]])
        comp_poc_dict[company_entries[j][0]].append(person_of_contact_entries[j][0])
    else:
        comp = random.sample(company_entries)
        poc_company_entries.append([comp[0],person_of_contact_entries[j][0]])
        comp_poc_dict[comp[0]].append(person_of_contact_entries[j][0])

```

(Note: To ensure the number of POC entries generated are greater than company entries we added this check)

```

if len(company_entries) < len(person_of_contact_entries):
    raise ValueError("Less number of POC than required")

```

- To connect POC and Opportunity, we took the same company Opportunity and POC and randomly assigned opportunity to POC

```

poc_company_entries = []
for j in range(len(person_of_contact_entries)):
    if j<len(company_entries):
        poc_company_entries.append([company_entries[j][0],person_of_contact_entries[j][0]])
        comp_poc_dict[company_entries[j][0]].append(person_of_contact_entries[j][0])
    else:
        comp = random.sample(company_entries)
        poc_company_entries.append([comp[0],person_of_contact_entries[j][0]])
        comp_poc_dict[comp[0]].append(person_of_contact_entries[j][0])

```

Here is the complete code for making the below :

```

opportunity_poc_entries = []
comp_opp_dict = {}
comp_poc_dict = {}
for comp in company_entries:
    comp_opp_dict[comp[0]] = []
    comp_poc_dict[comp[0]] = []

opportunity_company_entries = []
for j in range(len(opportunity_entries)):
    comp_id = random.sample(company_entries,1)[0]
    opportunity_company_entries.append([comp[0],opportunity_entries[j][0]])
    comp_opp_dict[comp[0]].append(opportunity_entries[j][0])

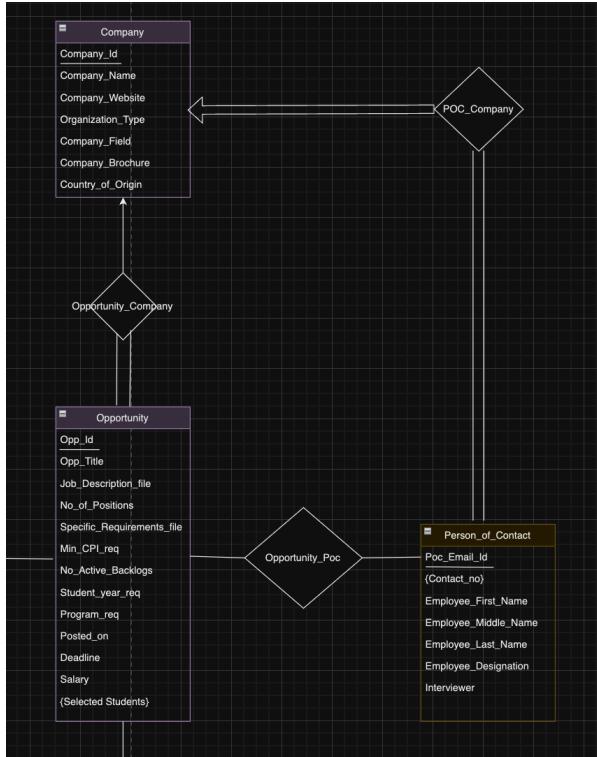


poc_company_entries = []
for j in range(len(person_of_contact_entries)):
    if j<len(company_entries):
        poc_company_entries.append([company_entries[j][0],person_of_contact_entries[j][0]])
        comp_poc_dict[company_entries[j][0]].append(person_of_contact_entries[j][0])
    else:
        comp = random.sample(company_entries)
        poc_company_entries.append([comp[0],person_of_contact_entries[j][0]])
        comp_poc_dict[comp[0]].append(person_of_contact_entries[j][0])



for comp,opps in comp_opp_dict.items():
    pocs = comp_poc_dict[comp]
    for opp in opps:
        k = random.randint(1,4)
        poc = random.sample(pocs,min(k,len(pocs)-1))
        opportunity_poc_entries.append([opp,poc])

```



Here is the link for generated data : [Link](#)

INDEXING

2. Please explain and implement the indexing over one of the columns (where the search needs to be optimized), user-defined data types, and table extensions.

- We've performed indexing on schemas to facilitate extensive querying. This indexing aims to slash query times and enhance the overall efficiency of our database system. We've selected a set of frequently queried attributes within our database to serve as our indices.

Factors on which we selected indexing attribute-

1. Most accessed attributes:

- Attributes that are frequently accessed in queries, especially in WHERE, ORDER BY, and GROUP BY clauses should be considered for indexing. This is because an index on these columns will speed up data retrieval.
- Some examples include columns used to filter rows like status, type, date columns used to sort output like created_at, id columns used to group aggregation functions like user_id, category_id.

2. Attributes used most as foreign keys in other tables:

- Columns which are often used as foreign keys to join multiple tables should be indexed. Foreign key columns are queried frequently during joins. Indexes will make these joins faster.
- For example, if a 'user_id' column exists in many tables, it should be indexed since WHERE clauses are likely to filter on 'user_id' during joins.

3. Columns declared as UNIQUE:

- Unique columns are good candidates for indexing. Unique indexes enforce uniqueness and also speed up lookups of individual rows.

Indexing Results

1. On student table

Query = SELECT * FROM Student WHERE Student_ID<2000;
Time required before indexing : 0.0039s

Indexing => CREATE INDEX student_idx ON Student(Student_ID,Department);
Time required after indexing : 0.0013s

2. Company

SELECT * FROM Company WHERE Company_Field="Tech";
Time required before indexing : 0.00062s

```
CREATE INDEX company_indx ON Company(Company_ID,Company_Field);  
Time required after indexing : 0.00048s
```

3. Opportunity

```
SELECT * FROM Opportunity WHERE Salary>=65000;  
Time required before indexing: 0.0049s
```

```
CREATE INDEX opp_indx ON Opportunity(Opportunity_ID,Salary);  
Time required after indexing: 0.0012s
```

4. Placement

```
SELECT * FROM Placement WHERE Company_Name="IBM";  
Time required before indexing: 0.0076s
```

```
CREATE INDEX plac_indx ON Placement(Placement_Id,Company_Name);  
Time required after indexing: 0.0010s
```

5. Academic_Info

```
SELECT * FROM Academic_Info WHERE CPI>=8.0;  
Time required before indexing: 0.0056s
```

```
CREATE INDEX acad_indx ON Academic_Info(Academic_Info_ID,CPI);  
Time required after indexing: 0.0019s
```

6. Person Of Contact

```
SELECT * FROM Person_of_Contact WHERE Employee_Designation="Director";  
Time required before indexing: 0.00083s
```

```
CREATE INDEX poc_indx ON Person_of_Contact(Poc_Email_ID);  
Time required after indexing: 0.00063s
```

User-Defined Data Type:

The MySQL implementation lacks support for user-defined data types. Nevertheless, we've utilized ENUM to designate certain attributes that can only accept specific values.

1. Gender ENUM('Male', 'Female', 'Other') NOT NULL : (in table 'student')

The gender column in the student table is defined as an ENUM data type that restricts the values allowed to only 'Male', 'Female', or 'Other'. This provides domain constraints to ensure standardized gender data compared to using an open VARCHAR or TEXT type.

The ENUM specifies that any record inserted or updated in the student table can only have a gender value of 'Male', 'Female', or 'Other'. No other gender values can be provided.

In addition, the NOT NULL constraint requires that a gender value must be provided and cannot be left as NULL. This ensures that every student record has an associated gender value defined as one of the permitted ENUM values.

Overall, the ENUM data type and NOT NULL constraint standardize the gender data while also making sure it is validated against a predefined set of allowed string values. This supports data integrity and uniformity.

2. Organization_Type ENUM('Public', 'Private', 'Non-Profit') NOT NULL : (in table 'company')

The Organization_Type column in the company table is defined as an ENUM data type that restricts the values allowed to only 'Public', 'Private', or 'Non-Profit'. This standardizes the types of organizations that can be specified in this system.

The ENUM enforces that company records can have an Organization_Type value of either 'Public', 'Private', or 'Non-Profit'. No other organizational type values are allowed through this data definition.

Table Extensions

Key reasons why table extensions can be useful in database design-

1. Support Evolving Requirements

- a. Table extensions allow the database model to gracefully evolve over time as new requirements and attributes emerge.
- b. Instead of continually altering existing tables which can be disruptive, new columns can simply be added to extensions.
- c. The utilization of table extensions for the serves to enhance the database's performance, uphold data consistency, and facilitate efficient organization and management of data.

2. Avoid Massive Wide Tables

- a. As requirements change, tables often keep having more and more columns added.
- b. This leads to very wide, cumbersome tables that get difficult to maintain.
Extensions let you keep the core table lean.

Let's say once we build the database, but then we feel the following requirements:

1. To separate the data of students on the basis of their department, we use table extension

```

CREATE TABLE MATH as (select * from student where Department = 'Mathematics');
CREATE TABLE ECO as (select * from student where Department = 'Economics');
CREATE TABLE PHY as (select * from student where Department = 'Physics');

```

assignment 2 - Warning - not supported

Query 1

```

126 • SELECT * FROM Academic_Info WHERE CPI>=8.0;
127 • CREATE INDEX acad_idx ON Academic_Info(Academic_Info_ID,CPI);
128
129 • CREATE TABLE MATH as (select * from student where Department = 'Mathematics');
130 • SELECT * from MATH;
131 • CREATE TABLE ECO as (select * from student where Department = 'Economics');
132 • CREATE TABLE PHY as (select * from student where Department = 'Physics');
133
134
135
136

```

Result Grid

Student_ID	Student_First_Na...	Student_Middle_Na...	Student_Last_Na...	Department	Active_Backlog	Gender	Year	Student_Image	Minors	Student_Email_Id	Contact_number
26	Sarah		Lee	Mathematics	3	Other	3	student_image_26.jpg	NULL	sarah.lee@iltgn.ac.in	+1-961-580-4948
138	Sarah	Olivia	Lee	Mathematics	3	Other	3	student_image_138.jpg	NULL	sarah.lee@iltgn.ac.in	+1-314-949-1239
203	John	Olivia	Smith	Mathematics	2	Male	1	student_image_203.jpg	Sociology	john.smith@iltgn.ac.in	+1-353-483-7783
282	Bob		Smith	Mathematics	2	Female	1	student_image_282.jpg	Art	bob.smith@iltgn.ac.in	+1-795-894-5118
314	Emily	William	Smith	Mathematics	0	Female	2	student_image_314.jpg	History	emily.smith@iltgn.ac.in	+1-409-203-8313
516	Sarah	Jacob	Lee	Mathematics	4	Male	2	student_image_516.jpg	NULL	sarah.lee@iltgn.ac.in	+1-539-250-3462
530	Bob	Jacob	Doe	Mathematics	4	Female	4	student_image_530.jpg	History	bob.doe@iltgn.ac.in	+1-738-515-9120
636	Michael	Olivia	Lee	Mathematics	3	Female	3	student_image_636.jpg	NULL	michael.lee@iltgn.ac.in	+1-380-561-5866
680	Jane	Olivia	Brown	Mathematics	2	Female	1	student_image_680.jpg	NULL	jane.brown@iltgn.ac.in	+1-585-561-3633
690	Bob		Davis	Mathematics	4	Female	3	student_image_690.jpg	NULL	bob.davis@iltgn.ac.in	+1-320-901-3116
785	David	Michael	Johnson	Mathematics	3	Female	4	student_image_785.jpg	NULL	david.johnson@iltgn.ac.in	+1-920-41-3995
866	Michael		Johnson	Mathematics	0	Female	4	student_image_866.jpg	English	michael.johnson@iltgn.ac.in	+1-470-730-3140
970	Eddy	Elizabeth	Lee	Mathematics	0	Female	1	student_image_970.jpg	Sociology	jenny.lee@iltgn.ac.in	+1-358-410-2423
1023	Jane	William	Lee	Mathematics	1	Male	1	student_image_1023.jpg	NULL	jane.lee@iltgn.ac.in	+1-730-953-8851
1070	Sarah	Olivia	Doe	Mathematics	1	Female	3	student_image_1070.jpg	NULL	sarah.doe@iltgn.ac.in	+1-750-891-9524
1135	John		Brown	Mathematics	3	Other	4	student_image_1135.jpg	History	john.brown@iltgn.ac.in	+1-325-896-6751
1184	David	Olivia	Smith	Mathematics	2	Male	3	student_image_1184.jpg	Econom.	christopher.smith@iltgn.ac.in	+1-932-499-6920
1263	Sarah	William	Doe	Mathematics	4	Other	2	student_image_1263.jpg	History	sarah.doe@iltgn.ac.in	+1-263-653-1278
1341	Emily	Jacob	Davis	Mathematics	1	Male	1	student_image_1341.jpg	NULL	emily.davis@iltgn.ac.in	+1-259-319-1540
1561	Jane	William	Brown	Mathematics	4	Female	3	student_image_1561.jpg	Biology	jane.brown@iltgn.ac.in	+1-901-346-3488
1754	John		Davis	Mathematics	0	Male	4	student_image_1754.jpg	English	john.davis@iltgn.ac.in	+1-530-563-8233
1777	Sarah	Olivia	Brown	Mathematics	0	Male	1	student_image_1777.jpg	NULL	sarah.brown@iltgn.ac.in	+1-924-329-7194
1802	Michael	Elizabeth	Lee	Mathematics	2	Other	4	student_image_1802.jpg	Art	michael.lee@iltgn.ac.in	+1-519-699-8449
1883	Sarah	Elizabeth	Doe	Mathematics	0	Male	1	student_image_1883.jpg	Chemistry	sarah.doe@iltgn.ac.in	+1-995-408-7985

MATH 89

Query Completed

Read Only

2. Also we can create separate tables from student table on the basis of gender
 - a. Male Students

assignment 2 - Warning - not supported

Query 1

```

168 • CREATE TABLE PHY as (select * from student where Department = 'Physics');
169
170 • CREATE TABLE M_STU as (select * from student where Gender="Male");
171 • SELECT * FROM M_STU;
172

```

Result Grid

Student_ID	Student_First_Na...	Student_Middle_Na...	Student_Last_Na...	Department	Active_Backlog	Gender	Year	Student_Image	Minors	Student_Email_Id	Contact_number
64	Michael	Jacob	Brown	Sociology	3	Male	4	student_image_64.jpg	History	michael.brown@itgn.ac.in	+1-566-468-7844
93	Bob	William	Johnson	English	3	Male	2	student_image_93.jpg	Chemistry	bob.johnson@itgn.ac.in	+1-387-984-9022
100	Sarah	Jacob	Smith	Psychology	2	Male	4	student_image_100.jpg	Biology	sarah.smith@itgn.ac.in	+1-271-605-4152
110	John	Hull	Doe	Biology	2	Male	2	student_image_110.jpg	Psychology	john.doe@itgn.ac.in	+1-633-847-8573
114	Michael	William	Davis	English	4	Male	3	student_image_114.jpg	Biology	michael.davis@itgn.ac.in	+1-602-508-6941
120	Emily	Hull	Smith	Chemistry	2	Male	2	student_image_120.jpg	English	emily.smith@itgn.ac.in	+1-216-210-9464
130	Michael	Elizabeth	Lee	Psychology	3	Male	3	student_image_130.jpg	Mathematics	michael.lee@itgn.ac.in	+1-123-472-5630
200	Jane	Hull	Smith	Physics	1	Male	4	student_image_200.jpg	Biology	jane.smith@itgn.ac.in	+1-946-172-6128
203	John	Olivia	Smith	Mathematics	2	Male	1	student_image_203.jpg	Sociology	john.smith@itgn.ac.in	+1-353-463-7783
206	Jane	Elizabeth	Brown	Sociology	3	Male	4	student_image_206.jpg	HULL	jane.brown@itgn.ac.in	+1-929-719-9747
223	Emily	Elizabeth	Brown	Economics	0	Male	2	student_image_223.jpg	Chemistry	emily.brown@itgn.ac.in	+1-377-377-6670
241	John	Hull	Brown	English	2	Male	2	student_image_241.jpg	Economics	john.brown@itgn.ac.in	+1-858-733-3717
312	John	Olivia	Doe	Psychology	0	Male	2	student_image_312.jpg	Chemistry	john.doe@itgn.ac.in	+1-121-376-2059
382	Michael	Hull	Smith	Biology	4	Male	2	student_image_382.jpg	Mathematics	michael.smith@itgn.ac.in	+1-996-294-1915
403	Sarah	Olivia	Doe	Physics	4	Male	3	student_image_403.jpg	HULL	sarah.doe@itgn.ac.in	+1-173-736-7428
474	John	Olivia	Davis	Biology	4	Male	4	student_image_474.jpg	English	john.davis@itgn.ac.in	+1-182-387-2271
481	Sarah	Olivia	Smith	Economics	2	Male	3	student_image_481.jpg	HULL	sarah.smith@itgn.ac.in	+1-962-215-2450

M_STU 91

Action Output

Time	Action	Response	Duration / Fetch Time
369 13:52:26	DROP TABLE 'student', 'Application'	0 row(s) affected	0.035 sec
370 13:52:29	CREATE TABLE Application (Student_ID INT, Opp_ID INT, Resume_File VARCHAR(255) NOT NULL, Resume_File_Name VARCHAR(255), Status V...	0 row(s) affected	0.019 sec
371 13:52:42	SHOW SESSION VARIABLES LIKE 'lower_case_table_names'	OK	0.000 sec
372 13:52:42	SHOW DATABASES	OK	0.000 sec
373 13:52:47	SHOW SESSION VARIABLES LIKE 'lower_case_table_names'	OK	0.000 sec
374 13:52:47	SHOW COLUMNS FROM 'student', 'application'	OK	0.000 sec
375 13:52:50	PREPARE stmt FROM 'INSERT INTO 'student', 'application' ('Student_ID', 'Opp_ID', 'Resume_File', 'Resume_File_Name', 'Status', 'Round_Number_Reache...	OK	0.000 sec
376 13:52:56	DEALLOCATE PREPARE stmt	OK	0.000 sec
377 14:03:55	CREATE TABLE M_STU as (select * from student where Gender="Male")	329 row(s) affected Records: 329 Duplicates: 0 War...	0.111 sec
378 14:04:00	SELECT * FROM M_STU LIMIT 0, 1000	329 row(s) returned	0.0035 sec / 0.00081...

Query Completed

b. Female Students

assignment 2 - Warning - not supported

Query 1

```

171 • SELECT * FROM M_STU;
172
173
174 • CREATE TABLE F_STU as (select * from student where Gender="Female");
175 • SELECT * FROM F_STU;
176

```

Result Grid

Student_ID	Student_First_Na...	Student_Middle_Na...	Student_Last_Na...	Department	Active_Backlog	Gender	Year	Student_Image	Minors	Student_Email_Id	Contact_number
38	Jane	Hull	Davis	Biology	4	Female	4	student_image_38.jpg	Chemistry	jane.davis@itgn.ac.in	+1-848-693-2921
58	John	William	Johnson	English	1	Female	4	student_image_58.jpg	Biology	john.johnson@itgn.ac.in	+1-517-824-9255
68	Bob	Olivia	Doe	Biology	1	Female	1	student_image_68.jpg	English	bob.doe@itgn.ac.in	+1-847-360-9347
70	Michael	Olivia	Smith	English	4	Female	1	student_image_70.jpg	Economics	michael.smith@itgn.ac.in	+1-538-922-7707
152	Alice	Jacob	Lee	Physics	2	Female	3	student_image_152.jpg	English	alice.lee@itgn.ac.in	+1-374-524-1874
154	Michael	Hull	Davis	Biology	3	Female	1	student_image_154.jpg	Chemistry	michael.davis@itgn.ac.in	+1-339-632-6648
202	John	Elizabeth	Smith	Physics	2	Female	2	student_image_202.jpg	Chemistry	john.smith@itgn.ac.in	+1-370-790-3354
210	Michael	Hull	Brown	Physics	3	Female	4	student_image_210.jpg	History	michael.brown@itgn.ac.in	+1-471-109-7489
231	Emily	Hull	Johnson	Art	4	Female	4	student_image_231.jpg	Physics	emily.johnson@itgn.ac.in	+1-918-885-5338
255	David	Olivia	Lee	English	2	Female	2	student_image_255.jpg	Biology	david.lee@itgn.ac.in	+1-318-427-6665
282	Bob	Hull	Smith	Mathematics	2	Female	1	student_image_282.jpg	Art	bob.smith@itgn.ac.in	+1-795-894-5118
314	Emily	William	Smith	Mathematics	0	Female	2	student_image_314.jpg	History	emily.smith@itgn.ac.in	+1-409-203-8373
436	David	Elizabeth	Lee	Biology	2	Female	2	student_image_436.jpg	HULL	david.lee@itgn.ac.in	+1-993-226-8676
494	Sarah	Hull	Lee	Physics	3	Female	3	student_image_494.jpg	Economics	sarah.lee@itgn.ac.in	+1-846-690-9464
517	Emily	Hull	Brown	Physics	2	Female	3	student_image_517.jpg	History	emily.brown@itgn.ac.in	+1-572-776-7961
530	Bob	Jacob	Doe	Mathematics	4	Female	4	student_image_530.jpg	History	bob.doe@itgn.ac.in	+1-738-515-9120
548	Linh	Olivia	Brown	Chemistry	0	Female	3	student_image_548.jpg	HULL	linh.brown@itgn.ac.in	+1-874-240-7929

F_STU 92

Action Output

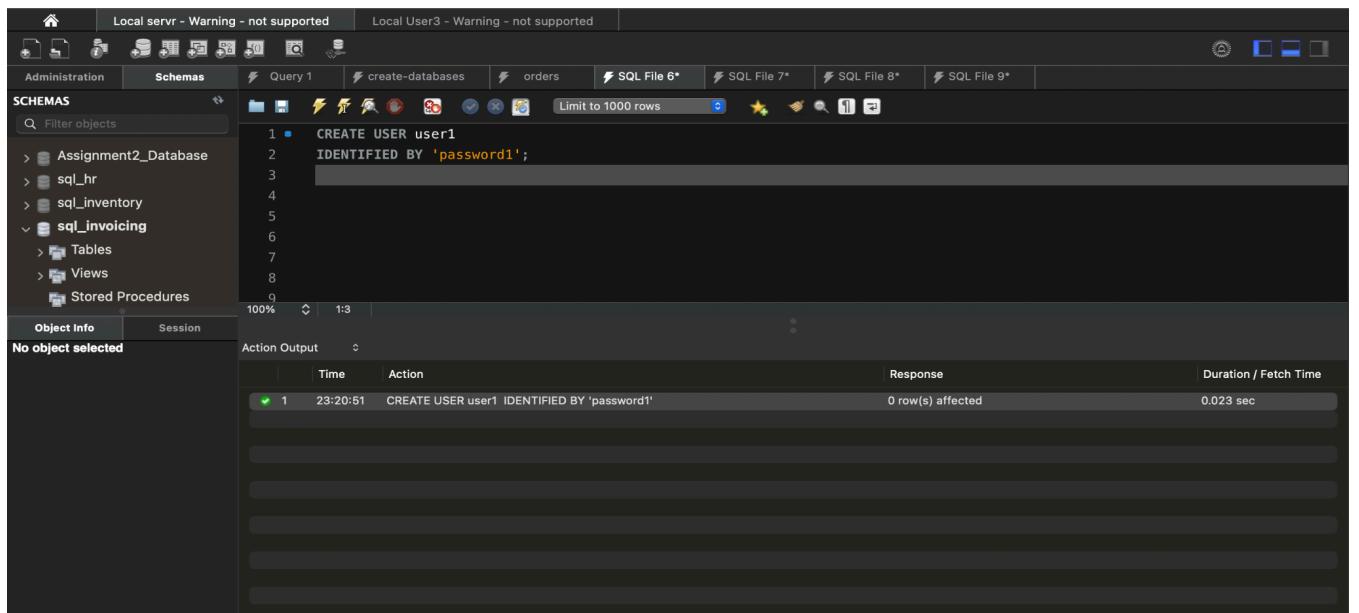
Time	Action	Response	Duration / Fetch Time
371 13:52:42	SHOW SESSION VARIABLES LIKE 'lower_case_table_names'	OK	0.000 sec
372 13:52:42	SHOW DATABASES	OK	0.000 sec
373 13:52:47	SHOW SESSION VARIABLES LIKE 'lower_case_table_names'	OK	0.000 sec
374 13:52:47	SHOW COLUMNS FROM 'student', 'application'	OK	0.000 sec
375 13:52:50	PREPARE stmt FROM 'INSERT INTO 'student', 'application' ('Student_ID', 'Opp_ID', 'Resume_File', 'Resume_File_Name', 'Status', 'Round_Number_Reache...	OK	0.000 sec
376 13:52:56	DEALLOCATE PREPARE stmt	OK	0.000 sec
377 14:03:55	CREATE TABLE M_STU as (select * from student where Gender="Male")	329 row(s) affected Records: 329 Duplicates: 0 War...	0.111 sec
378 14:04:00	SELECT * FROM M_STU LIMIT 0, 1000	329 row(s) returned	0.0035 sec / 0.00081...
379 14:05:46	CREATE TABLE F_STU as (select * from student where Gender="Female")	334 row(s) affected Records: 334 Duplicates: 0 War...	0.030 sec
380 14:05:51	SELECT * FROM F_STU LIMIT 0, 1000	334 row(s) returned	0.00091 sec / 0.0002...

Query Completed

13.2 Responsibility of G2:

1. Create a user named "user1" with the password "password1"

To create a new user in the database, we access the database using the root and create a new user.



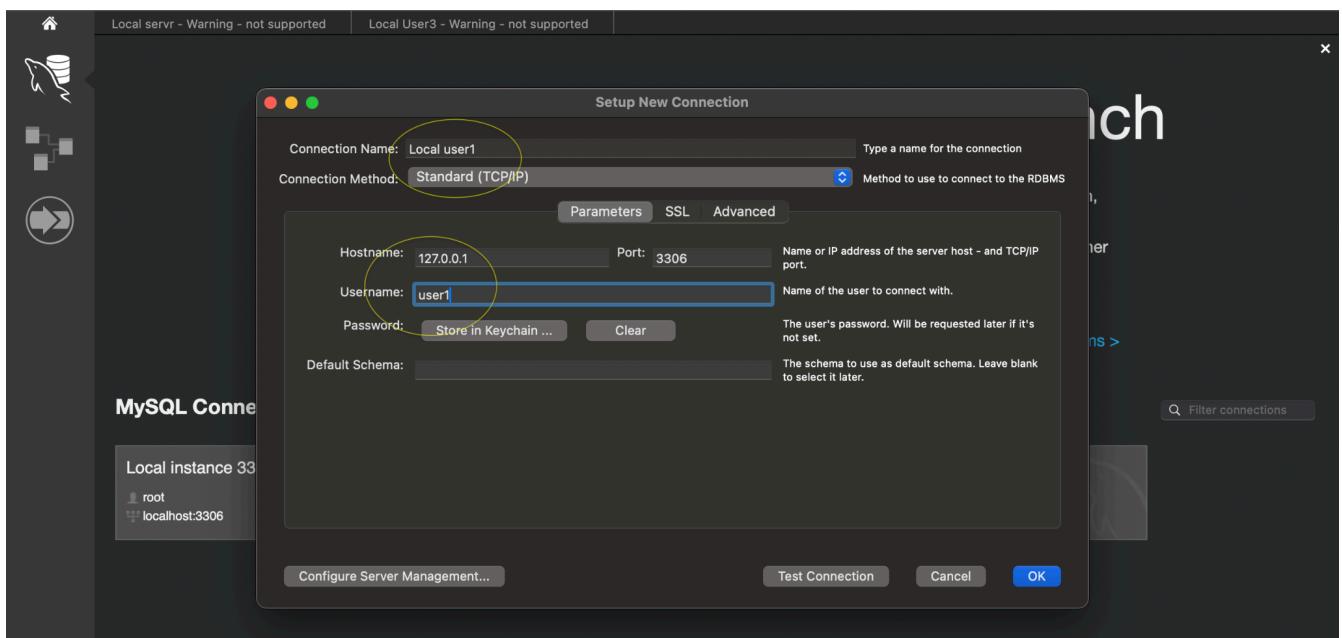
The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' tree view is open, showing databases like 'Assignment2_Database', 'sql_hr', 'sql_inventory', and 'sql_invoicing'. The 'sql_invoicing' database is selected. On the right, the main pane displays a query editor with the following SQL code:

```
CREATE USER user1
IDENTIFIED BY 'password1';
```

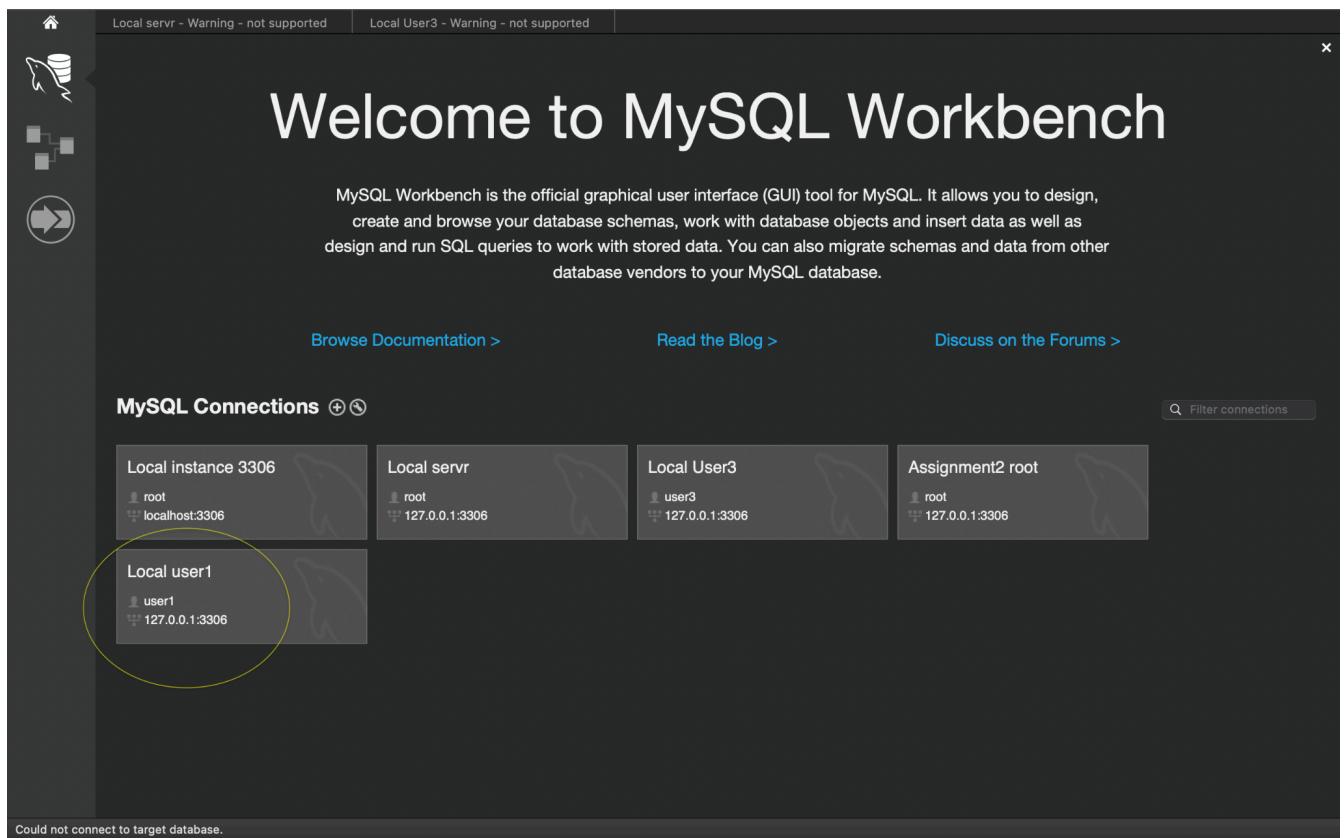
Below the query editor, the 'Action Output' section shows the execution results:

Action	Time	Response	Duration / Fetch Time
CREATE USER user1 IDENTIFIED BY 'password1'	23:20:51	0 row(s) affected	0.023 sec

This creates a new user "user1", but we have to activate a connection.



And with this, we have created a new user, which we can access with our password.



- 2.** Create Views on any of the two tables formed by G1 as view1 and view2. And make sure that views contain columns from at least two tables and one additional column with the user-defined data type

VIEW1

This view is created by using the placement and Opportunity entity sets. This view does not have an enum value, as the new one have and this one has interesting attributes in it.

Here, we have provided the code and how the View looks like when selected.

From the Placement Entity, it has:- Placement_ID, Placement_Method, Company_Name, Job_Designation

From the Opportunity Entity it has:- Opp_Title, Min_CPI_req

It is a joint view, where we select those columns that have same Opportunity Title, and Placement Job Description and add them.

The screenshot shows the SQL Server Management Studio interface. In the Object Explorer, under 'Assignment2_Database' > 'Tables', there is a 'view1' entry. The 'Script' button next to it is highlighted. The script pane shows the following T-SQL code:

```

CREATE VIEW view1 AS
SELECT placement.Placement_ID, placement.Placement_Method, Placement.Company_Name ,Placement.Job_Designation,
       Opportunity.Opp_Title, Opportunity.Min_CPI_req
FROM placement
JOIN Opportunity ON Opportunity.Opp_Title = Placement.Job_Designation;

```

Below the script, a 'Result Grid' shows the data returned by the query:

Placement_ID	Placement_Meth...	Company_Name	Job_Designation	Opp_Title	Min_CPI_req
10	Campus Placement	Tesla	Systems Analyst	Systems Analyst	7.4
10	Campus Placement	Tesla	Systems Analyst	Systems Analyst	7.2
10	Campus Placement	Tesla	Systems Analyst	Systems Analyst	7.2
10	Campus Placement	Tesla	Systems Analyst	Systems Analyst	7.5
20	Direct Application	Finance Gurus	Software Engineer	Software Engineer	7.6
20	Direct Application	Finance Gurus	Software Engineer	Software Engineer	7.3
20	Direct Application	Finance Gurus	Software Engineer	Software Engineer	7.0
20	Direct Application	Finance Gurus	Software Engineer	Software Engineer	7.4
42	Campus Placement	Biz Geniuses	Graphic Designer	Graphic Designer	7.3
42	Campus Placement	Biz Geniuses	Graphic Designer	Graphic Designer	7.5
42	Campus Placement	Biz Geniuses	Graphic Designer	Graphic Designer	7.4
42	Campus Placement	Biz Geniuses	Graphic Designer	Graphic Designer	7.3
42	Campus Placement	Biz Geniuses	Graphic Designer	Graphic Designer	7.4
42	Campus Placement	Biz Geniuses	Graphic Designer	Graphic Designer	7.1
96	Direct Application	IBM	Software Tester	Software Tester	7.3
96	Direct Application	IBM	Software Tester	Software Tester	7.4

The status bar at the bottom right indicates 'Read Only'.

VIEW2

This view is created by using the Student and Opportunity entity sets.

This view does has an user-specified data type (given by enum value), It is Student.Gender, which can be Male, Female, Other.

Here, we have provided the code and what the View looks like when selected.

From the Student Entity, it has:- student.student_ID, student.Department, student.Gender ,student.Year,

From the Opportunity Entity it has:- Opportunity.Student_year_req, Opportunity.Program_req

It is a joint view, where we select those columns that have same Opportunity Student_year_req, and Student year and add them.

```
CREATE VIEW view2 AS
SELECT student.student_ID, student.Department, student.Gender ,student.Year,
Opportunity.Student_year_req, Opportunity.Program_req
FROM student
JOIN Opportunity ON Opportunity.Student_year_req = student.Year;
select * from view2
```

student_ID	Department	Gender	Year	Student_year_req	Program_req
15	Biology	Male	3	3	SE
15	Biology	Male	3	3	IT
15	Biology	Male	3	3	CS
15	Biology	Male	3	3	SE
15	Biology	Male	3	3	Graphic
15	Biology	Male	3	3	UI/UX
15	Biology	Male	3	3	DS
15	Biology	Male	3	3	CS
15	Biology	Male	3	3	UI/UX
15	Biology	Male	3	3	BA
16	Physics	Female	3	3	SE
16	Physics	Female	3	3	SE
16	Physics	Female	3	3	DS
16	Physics	Female	3	3	UI/UX
16	Physics	Female	3	3	Marketing
16	Physics	Female	3	3	CS
16	Physics	Female	3	3	DS
16	Physics	Female	3	3	UI/UX
16	Physics	Female	3	3	Graphic

3. Grant "user1" the following permissions on "table1": SELECT, UPDATE, DELETE

To grant the new user "user1" permissions, we will go back to the root connection, and from there, update and give the permissions for the new user

Table1

Here, we will be giving these permissions to the User1 for the "table1", which we have selected to be the opportunity table.

The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'Schemas' tab, the 'Assignment2_Database' is selected. In the main query editor window, the following SQL code is run:

```

1 • use Assignment2_Database;
2 • GRANT SELECT, UPDATE, DELETE on Opportunity to user1;
3 • show grants for user1;
4

```

The results grid shows the grants for user1:

Grants for user1@%	
GRANT USAGE ON *.* TO 'user1'@'%'	
GRANT SELECT, UPDATE, DELETE ON `assignment2`.`opportunity` TO 'user1'@'%'	
GRANT SELECT, UPDATE, DELETE ON `assignment2`.`view2` TO 'user1'@'%'	
GRANT SELECT, UPDATE, DELETE ON `assignment2`.`view1` TO 'user1'@'%'	

The action output pane shows the execution log with the following entries:

Action	Time	Response	Duration / Fetch Time
select * from Academic_Info LIMIT 0,1000	23:55:50	998 row(s) returned	0.0034 sec / 0.0008...
SELECT * FROM student LIMIT 0,1000	23:58:43	957 row(s) returned	0.0022 sec / 0.003...
select * from Opportunity LIMIT 0,1000	23:59:02	74 row(s) returned	0.0013 sec / 0.0004...
CREATE VIEW view2 AS SELECT stude...	00:05:22	0 row(s) affected	0.012 sec
select * from view2 LIMIT 0,1000	00:05:22	1000 row(s) returned	0.0035 sec / 0.0003...
use Assignment2_Database	00:08:47	0 row(s) affected	0.0024 sec
GRANT SELECT, UPDATE, DELETE on...	00:08:47	0 row(s) affected	0.0078 sec
show grants for user1	00:08:47	2 row(s) returned	0.00087 sec / 0.0000...
CREATE VIEW view2 AS SELECT stude...	00:10:25	Error Code: 1050. Table 'view2' already exists	0.0015 sec
use Assignment2_Database	00:10:47	0 row(s) affected	0.00029 sec
GRANT SELECT, UPDATE, DELETE on...	00:10:47	0 row(s) affected	0.0041 sec
show grants for user1	00:10:47	3 row(s) returned	0.00031 sec / 0.0000...
use Assignment2_Database	01:17:36	0 row(s) affected	0.0014 sec
GRANT SELECT, UPDATE, DELETE on...	01:17:36	0 row(s) affected	0.011 sec
show grants for user1	01:17:36	4 row(s) returned	0.00094 sec / 0.000...

Query Completed

4. Grant "user1" the following permissions on "view1": SELECT

View1

Here, we will be giving these permissions to the User1 for the “view1”.

The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'Schemas' tab, the 'Assignment2_Database' is selected. In the main query editor window, the following SQL code is run:

```

1 • use Assignment2_Database;
2 • GRANT SELECT on view1 to user1;
3 • show grants for user1;
4

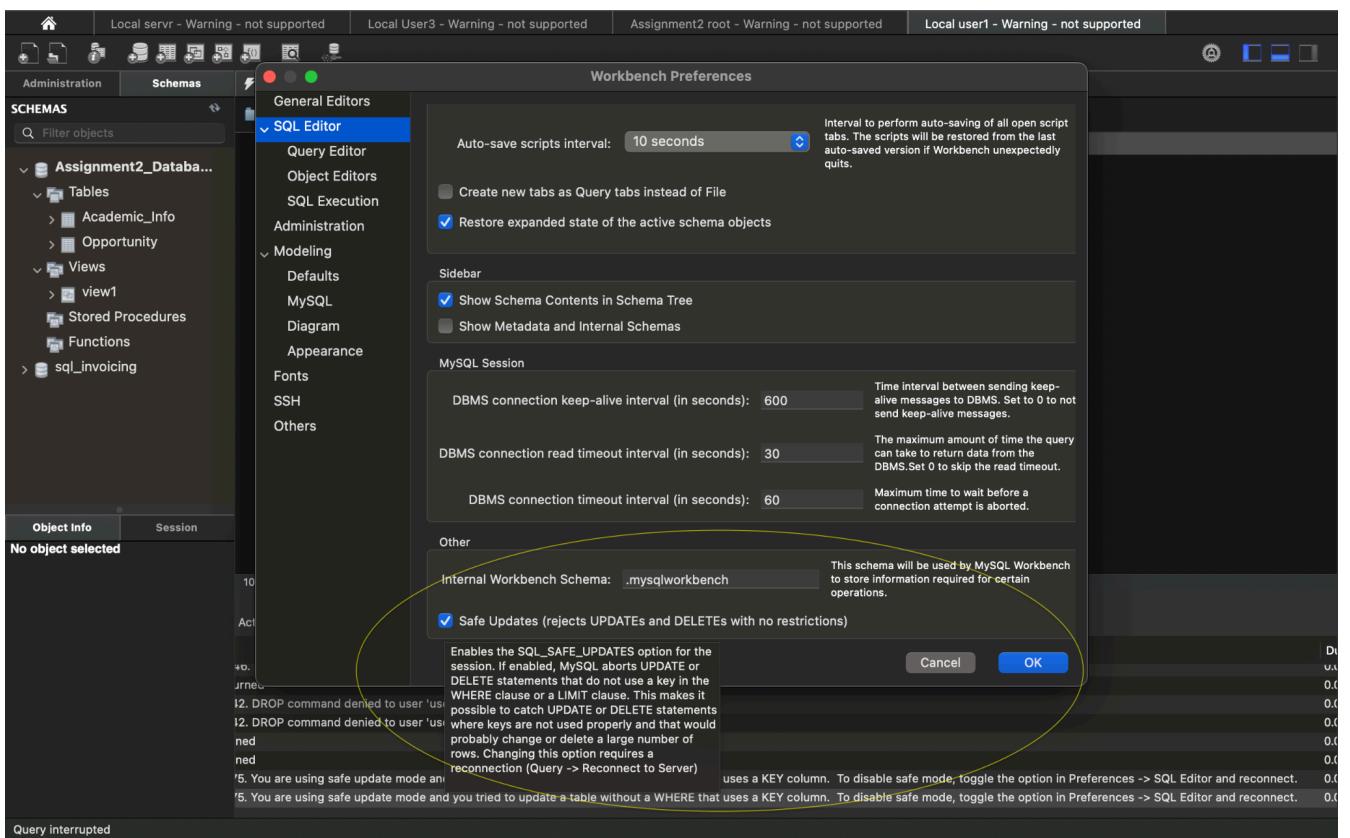
```

5. Try to perform *SELECT*, *UPDATE*, and *DELETE* operations on "table1" and "view1" as "user1" and report your findings.

TABLE1

Here, we will perform some operations on the DATA in table, but before that, We will change the setting of the safe update mode, where the Database disable editing without using a where clause.

This can be done by changing the preferences and disselecting safe update mode.



Now, we can see the Operations we perform:-

SELECT

The screenshot shows the MySQL Workbench interface with a query editor window. The query being run is:

```
Select * from Opportunity;
```

The results grid displays 16 rows of data from the Opportunity table. Below the results, the 'Action Output' section shows the history of operations performed, including SELECT statements and errors related to DROP TABLE and SELECT commands.

Opp_ID	Opp_Title	No_of_Positions	Specific_Requirements_...	Min_CPI_req	No_Active_Backlo...	Student_year_r...	Program_req	Job_Description_...	Posted_on
3	Graphic Designer	2	req_file_7.pdf	7.1	FALSE	2	Marketing	job_desc_14.pdf	2024-02-06 16:31:53.125
4	Software Tester	1	req_file_5.pdf	7.1	TRUE	3	UI/UX	job_desc_15.pdf	2024-02-02 16:31:53.125
5	Data Scientist	3	req_file_15.pdf	7.4	FALSE	4	IT	job_desc_15.pdf	2024-01-19 16:31:53.125
6	UX Designer	2	req_file_13.pdf	7.4	FALSE	3	CS	job_desc_13.pdf	2024-02-11 16:31:53.125
7	Graphic Designer	2	req_file_4.pdf	7.4	FALSE	2	CS	job_desc_9.pdf	2024-01-22 16:31:53.125
8	Web Developer	2	req_file_10.pdf	7.4	TRUE	3	DS	job_desc_4.pdf	2024-02-03 16:31:53.125
9	UI/UX Designer	1	req_file_15.pdf	7.4	TRUE	4	UI/UX	job_desc_2.pdf	2024-01-22 16:31:53.125
10	Systems Analyst	2	req_file_8.pdf	7.5	TRUE	4	UI/UX	job_desc_10.pdf	2024-01-21 16:31:53.125
11	Software Tester	5	req_file_8.pdf	7.4	TRUE	2	UI/UX	job_desc_5.pdf	2024-02-02 16:31:53.125
12	Data Analyst	1	req_file_1.pdf	7.5	FALSE	2	CS	job_desc_13.pdf	2024-02-03 16:31:53.125
13	Graphic Designer	4	req_file_11.pdf	7.3	FALSE	3	UI/UX	job_desc_1.pdf	2024-01-24 16:31:53.125
14	UX Designer	2	req_file_7.pdf	7.3	FALSE	2	IT	job_desc_3.pdf	2024-02-10 16:31:53.125
15	Software Engineer	4	req_file_9.pdf	7.4	TRUE	4	CS	job_desc_13.pdf	2024-02-12 16:31:53.125
16	Marketing Analyst	3	req_file_6.pdf	7.4	FALSE	2	SE	job_desc_4.pdf	2024-01-22 16:31:53.125

UPDATE

Here, we will update the value of the CPI to +0.5 than before, as one can clearly in the image.

BEFORE UPDATION

Opp_ID	Opp_Title	No_of_Positions	Specific_Requirements_...	Min_CPI_req	No_Active_Backlo...	Student_year_r...	Program_req	Job_Description_...	Posted_on
3	Graphic Designer	2	req_file_7.pdf	7.1	FALSE	2	Marketing	job_desc_14.pdf	2024-02-06 16:31:53.125
4	Software Tester	1	req_file_5.pdf	7.1	TRUE	3	UI/UX	job_desc_15.pdf	2024-02-02 16:31:53.125
5	Data Scientist	3	req_file_15.pdf	7.4	FALSE	4	IT	job_desc_15.pdf	2024-01-19 16:31:53.125
6	UX Designer	2	req_file_13.pdf	7.4	FALSE	3	CS	job_desc_13.pdf	2024-02-11 16:31:53.125
7	Graphic Designer	2	req_file_4.pdf	7.4	FALSE	2	CS	job_desc_9.pdf	2024-01-22 16:31:53.125
8	Web Developer	2	req_file_10.pdf	7.4	TRUE	3	DS	job_desc_4.pdf	2024-02-03 16:31:53.125

AFTER UPDATION

Opp_ID	Opp_Title	No_of_Positions	Specific_Requirements_...	Min_CPI_req	No_Active_Backlo...	Student_year_r...	Program_req	Job_Description_...	Posted_on
2	Financial Analyst	1	req_file_10.pdf	7.7	TRUE	3	BA	job_desc_10.pdf	2024-02-07 16:31:53.125
3	Graphic Designer	2	req_file_7.pdf	7.6	FALSE	2	Marketing	job_desc_14.pdf	2024-02-06 16:31:53.125
4	Software Tester	1	req_file_5.pdf	7.6	TRUE	3	UI/UX	job_desc_15.pdf	2024-02-02 16:31:53.125
5	Data Scientist	3	req_file_15.pdf	7.9	FALSE	4	IT	job_desc_15.pdf	2024-01-19 16:31:53.125
6	UX Designer	2	req_file_13.pdf	7.9	FALSE	3	CS	job_desc_13.pdf	2024-02-11 16:31:53.125
7	Graphic Designer	2	req_file_4.pdf	7.9	FALSE	2	CS	job_desc_9.pdf	2024-01-22 16:31:53.125
8	Web Developer	2	req_file_10.pdf	7.9	TRUE	3	DS	job_desc_4.pdf	2024-02-03 16:31:53.125

The code and screen for the updation process

The screenshot shows a database interface with the following details:

- Schemas:** Assignment2_Database (Tables: Academic_Info, Opportunity, Views, Stored Procedures, Functions, sql_invoicing).
- Query Editor:**

```

1 • Update opportunity set Min_CPI_req = 0.5 + Min_CPI_req;
2
3 • select * from opportunity
    
```
- Result Grid:** Shows a table of opportunities with columns: Opp_ID, Opp_Title, No_of_Positions, Specific_Requirements..., Min_CPI_req, No_Active_Backlog..., Student_year_r..., Program_req, Job_Description_..., and Posted_on.
- Action Output:** Displays the history of actions taken, including updates and selects, along with their execution time, response, and duration.

DELETION

In deletion, we are deleting the Opportunities which have their job description in "job_desc_15.pdf". The jobs with opp_id 4 and 5 have been deleted.

BEFORE DELETION

Opp_ID	Opp_Title	No_of_Positions	Specific_Requirements_...	Min_CPI_req	No_Active_Backlog...	Student_year_r...	Program_req	Job_Description_...
1	Data Analyst	5	req_file_15.pdf	7.8	FALSE	2	Graphic	job_desc_9.pdf
2	Financial Analyst	1	req_file_10.pdf	7.7	TRUE	3	BA	job_desc_8.pdf
3	Graphic Designer	2	req_file_7.pdf	7.6	FALSE	2	Marketing	job_desc_14.pdf
4	Software Tester	1	req_file_5.pdf	7.6	TRUE	3	UI/UX	job_desc_15.pdf
5	Data Scientist	3	req_file_15.pdf	7.9	FALSE	4	IT	job_desc_15.pdf
6	UX Designer	2	req_file_13.pdf	7.9	FALSE	3	CS	job_desc_13.pdf
7	Graphic Designer	2	req_file_4.pdf	7.9	FALSE	2	CS	job_desc_9.pdf
8	Web Developer	2	req_file_10.pdf	7.9	TRUE	3	DS	job_desc_4.pdf
9	UI/UX Designer	1	req_file_15.pdf	7.9	TRUE	4	UI/UX	job_desc_2.pdf
10	Systems Analyst	2	req_file_8.pdf	8.0	TRUE	4	UI/UX	job_desc_10.pdf
11	Software Tester	5	req_file_8.pdf	7.9	TRUE	2	UI/UX	job_desc_5.pdf
12	Data Analyst	1	req_file_1.pdf	8.0	FALSE	2	CS	job_desc_13.pdf

AFTER DELETION

Opp_ID	Opp_Title	No_of_Positions	Specific_Requirements_...	Min_CPI_req	No_Active_Backlog...	Student_year_r...	Program_req	Job_Description_...
1	Data Analyst	5	req_file_15.pdf	7.8	FALSE	2	Graphic	job_desc_9.pdf
2	Financial Analyst	1	req_file_10.pdf	7.7	TRUE	3	BA	job_desc_8.pdf
3	Graphic Designer	2	req_file_7.pdf	7.6	FALSE	2	Marketing	job_desc_14.pdf
6	UX Designer	2	req_file_13.pdf	7.9	FALSE	3	CS	job_desc_13.pdf
7	Graphic Designer	2	req_file_4.pdf	7.9	FALSE	2	CS	job_desc_9.pdf
8	Web Developer	2	req_file_10.pdf	7.9	TRUE	3	DS	job_desc_4.pdf

Here is the full code and screen for deletion process.

The screenshot shows the MySQL Workbench interface with the following details:

- Left Panel (Object Navigator):** Shows the schema **Assignment2_Database** with its tables (**Academic_Info**, **Opportunity**, **Views**), stored procedures, functions, and a sql_invoicing table.
- Central Panel (Query Editor):** Displays the SQL code used for the deletion:

```

1 • use Assignment2_Database;
2
3 • DELETE FROM Opportunity
4 WHERE Job_Description_file = 'job_desc_15.pdf';
5
6 • select * from opportunity
    
```
- Result Grid:** Shows the results of the `select * from opportunity` query, listing various job opportunities with columns like Opp_ID, Opp_Title, No_of_Positions, Specific_Requirements, Min_CPI_req, No_Active_Backlog, Student_year_r..., Program_req, Job_Description..., and Posted_on.
- Action Output:** Displays the history of actions taken during the session, including the successful deletion attempt and the subsequent error message.

Opp_ID	Opp_Title	No_of_Positions	Specific_Requirements	Min_CPI_req	No_Active_Backlog	Student_year_r...	Program_req	Job_Description...	Posted_on
1	Data Analyst	5	req_file_15.pdf	7.8	FALSE	2	Graphic	job_desc_9.pdf	2024-01-26 16:31:53.125
2	Financial Analyst	1	req_file_10.pdf	7.7	TRUE	3	BA	job_desc_8.pdf	2024-01-18 16:31:53.125
3	Graphic Designer	2	req_file_7.pdf	7.6	FALSE	2	Marketing	job_desc_14.pdf	2024-02-06 16:31:53.125
6	UX Designer	2	req_file_13.pdf	7.9	FALSE	3	CS	job_desc_13.pdf	2024-02-11 16:31:53.125
7	Graphic Designer	2	req_file_4.pdf	7.9	FALSE	2	CS	job_desc_9.pdf	2024-01-22 16:31:53.125
8	Web Developer	2	req_file_10.pdf	7.9	TRUE	3	DS	job_desc_4.pdf	2024-02-03 16:31:53.125
9	UI/UX Designer	1	req_file_15.pdf	7.9	TRUE	4	UI/UX	job_desc_2.pdf	2024-01-22 16:31:53.125
10	Systems Analyst	2	req_file_8.pdf	8.0	TRUE	4	UI/UX	job_desc_10.pdf	2024-01-21 16:31:53.125
11	Software Tester	5	req_file_8.pdf	7.9	TRUE	2	UI/UX	job_desc_5.pdf	2024-02-02 16:31:53.125
12	Data Analyst	1	req_file_1.pdf	8.0	FALSE	2	CS	job_desc_13.pdf	2024-02-03 16:31:53.125
13	Graphic Designer	4	req_file_11.pdf	7.8	FALSE	3	UI/UX	job_desc_1.pdf	2024-01-24 16:31:53.125
14	UX Designer	2	req_file_7.pdf	7.8	FALSE	2	IT	job_desc_3.pdf	2024-02-10 16:31:53.125
15	Software Engineer	4	req_file_9.pdf	7.9	TRUE	4	CS	job_desc_14.pdf	2024-02-12 16:31:53.125

Action	Time	Action	Response	Duration / Fetch Time
12	01:35:52	select * from opportunity LIMIT 0, 1000	74 row(s) returned	0.00061 sec / 0.0000...
13	01:37:17	use Assignment2_Database	0 row(s) affected	0.0021 sec
14	01:37:17	DELETE FROM Opportunity WHERE Job_Description_file...	Error Code: 1054. Unknown column 'job_desc_15.pdf' in 'where clause'	0.0030 sec
15	01:37:37	use Assignment2_Database	0 row(s) affected	0.00026 sec
16	01:37:37	DELETE FROM Opportunity WHERE Job_Description_file...	4 row(s) affected	0.0077 sec
17	01:37:37	select * from opportunity LIMIT 0, 1000	70 row(s) returned	0.00061 sec / 0.0000...

VIEW1

Here, we will perform the same operation, but on view1.

SELECT

The screenshot shows the MySQL Workbench interface. The left sidebar displays the schema 'Assignment2_Database' with its tables, views, and stored procedures. The main area contains a SQL editor window with the following content:

```
1 • select * from view1
```

The results are displayed in a grid titled 'Result Grid':

Placement_ID	Placement_Meth...	Company_Name	Job_Designation	Opp_Title	Min_CPI_req
10	Campus Placement	Tesla	Systems Analyst	Systems Analyst	7.9
10	Campus Placement	Tesla	Systems Analyst	Systems Analyst	7.7
10	Campus Placement	Tesla	Systems Analyst	Systems Analyst	7.7
10	Campus Placement	Tesla	Systems Analyst	Systems Analyst	8.0
20	Direct Application	Finance Gurus	Software Engineer	Software Engineer	8.1
20	Direct Application	Finance Gurus	Software Engineer	Software Engineer	7.8
20	Direct Application	Finance Gurus	Software Engineer	Software Engineer	7.5
20	Direct Application	Finance Gurus	Software Engineer	Software Engineer	7.9
42	Campus Placement	Biz Geniuses	Graphic Designer	Graphic Designer	7.8
42	Campus Placement	Biz Geniuses	Graphic Designer	Graphic Designer	8.0
42	Campus Placement	Biz Geniuses	Graphic Designer	Graphic Designer	7.9
42	Campus Placement	Biz Geniuses	Graphic Designer	Graphic Designer	7.8
42	Campus Placement	Biz Geniuses	Graphic Designer	Graphic Designer	7.9

The 'Object Info' and 'Session' panes are visible on the left, and the 'Result Grid' pane is on the right. The status bar at the bottom left says 'Query Completed'.

UPDATE

Here, we will update the value of the CPI to +0.5 than before, as one can see in the image,

```
1 • Update view1 set Min_CPI_req = 0.5 + Min_CPI_req;
2
3 • select * from view1
```

The updation fails, as we did not provide the user with the necessary permissions.

The screenshot shows the MySQL Workbench interface with the 'Action Output' pane open. The log entries are as follows:

Action	Time	Response
select * from view1 LIMIT 0, 1000	03:47:16	1000 row(s) returned
Update view1 set Min_CPI_req = 0.5 + Min_CPI_req;	03:47:40	Error Code: 1142. UPDATE command denied to user 'user1'@'localhost' for table 'view1'

DELETION

Here, we will update the value of the CPI to +0.5 than before, as one can see in the image,

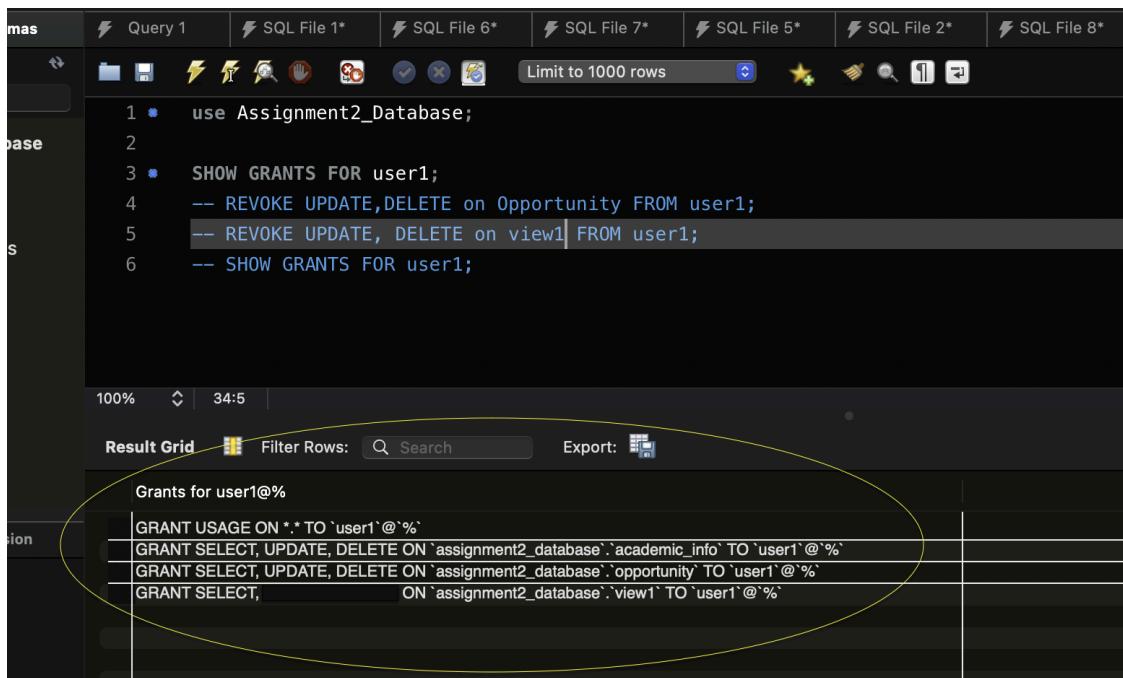
```
1 •    use Assignment2_Database;
2
3 •    DELETE FROM view1
4      WHERE Placement_Method = 'Direct Application';
5
6 •    select * from view1
```

The updation fails, as we did not provide the user with the necessary permissions.

Action	Output
Time	Action
38 03:50:16	use Assignment2_Database 0 row(s) affected
39 03:50:16	DELETE FROM view1 WH... Error Code: 1142. DELETE command denied to user 'user1'@'localhost' for table 'view1'

-
6. Revoke the UPDATE and DELETE permissions on "table1" for "user1" and report your findings.

BEFORE REVOKING THE PERMISSIONS:-



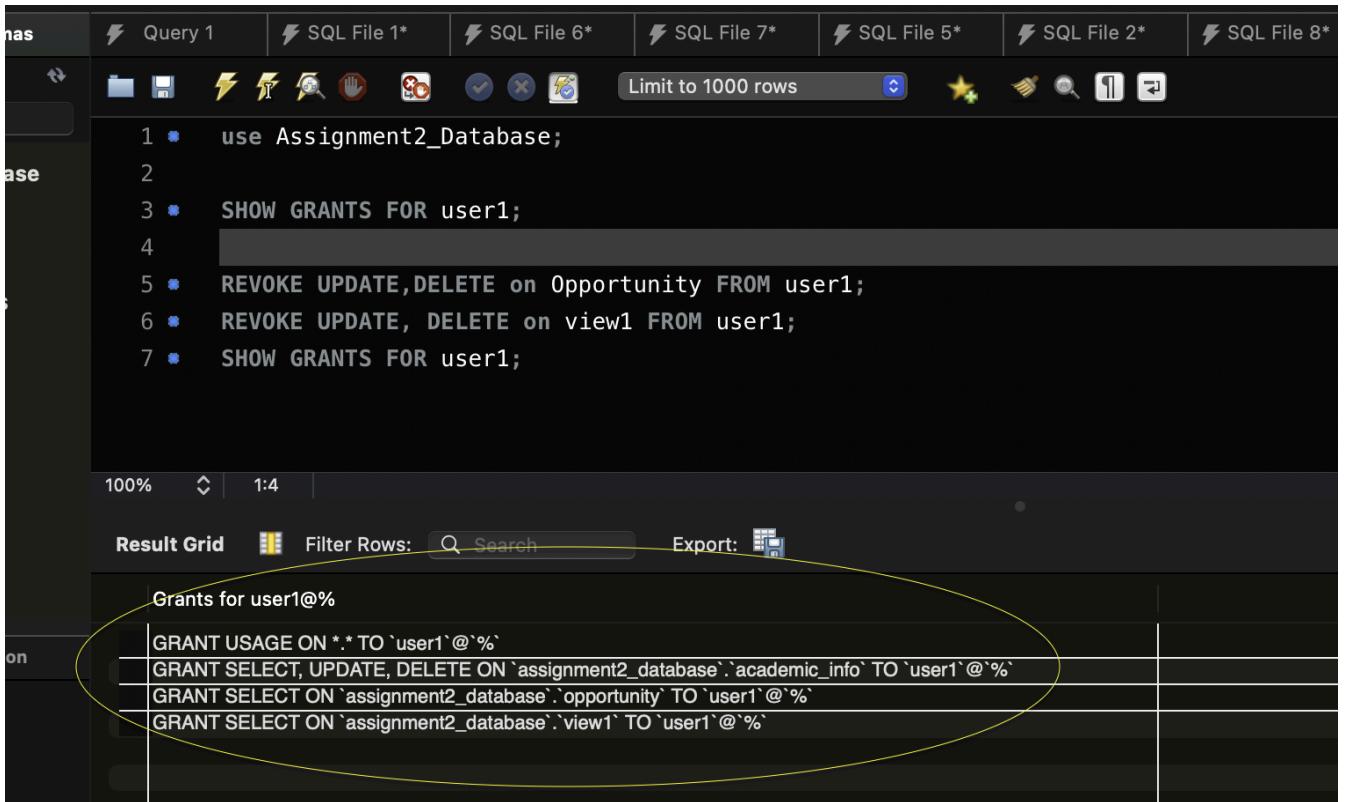
The screenshot shows the MySQL Workbench interface with a query editor and a results grid. The query editor contains the following SQL commands:

```
1 •    use Assignment2_Database;
2
3 •    SHOW GRANTS FOR user1;
4      -- REVOKE UPDATE,DELETE on Opportunity FROM user1;
5      -- REVOKE UPDATE, DELETE on view1 FROM user1;
6      -- SHOW GRANTS FOR user1;
```

The results grid displays the grants for user1@%:

Grants for user1@%
GRANT USAGE ON *.* TO `user1`@`%`
GRANT SELECT, UPDATE, DELETE ON `assignment2_database`.`academic_info` TO `user1`@`%`
GRANT SELECT, UPDATE, DELETE ON `assignment2_database`.`opportunity` TO `user1`@`%`
GRANT SELECT, ON `assignment2_database`.`view1` TO `user1`@`%`

AFTER REVOKING PERMISSIONS



The screenshot shows a MySQL Workbench interface with several tabs at the top: Query 1, SQL File 1*, SQL File 6*, SQL File 7*, SQL File 5*, SQL File 2*, and SQL File 8*. Below the tabs is a toolbar with various icons. A dropdown menu is open on the left, showing options like 'File', 'Edit', 'View', etc. The main area contains the following SQL code:

```
1 * use Assignment2_Database;
2
3 * SHOW GRANTS FOR user1;
4
5 * REVOKE UPDATE,DELETE on Opportunity FROM user1;
6 * REVOKE UPDATE, DELETE on view1 FROM user1;
7 * SHOW GRANTS FOR user1;
```

Below the code, the status bar shows '100%' and '1:4'. At the bottom, there are buttons for 'Result Grid' (highlighted with a yellow oval), 'Filter Rows', 'Search', and 'Export'.

The results grid shows the grants for user1@%:

Grants for user1@%
GRANT USAGE ON *.* TO `user1`@`%`
GRANT SELECT, UPDATE, DELETE ON `assignment2_database`.`academic_info` TO `user1`@`%`
GRANT SELECT ON `assignment2_database`.`opportunity` TO `user1`@`%`
GRANT SELECT ON `assignment2_database`.`view1` TO `user1`@`%`

(NOTE: The permission to Academic_Info table, was only put for testing purposes, please ignore it)

Also, to remove the new user "user1" 's permissions, we will go back to the root connection, and from there, revoke the permissions of the new user

-
7. Revoke the UPDATE and DELETE permissions on "table1" for "user1" and report your findings.

As expected, we are only able to perform SELECT, as that is the only permission remaining with the new user.

TABLE1

SELECT

The screenshot shows the MySQL Workbench interface with the following details:

- Left Panel:** Shows the database schema for Assignment2_Database, including Tables (Academic_Info, Opportunity), Views, Stored Procedures, Functions, and sql_invoicing.
- Query Editor:** Contains the SQL command: `1 • Select * from opportunity;`
- Result Grid:** Displays the data from the opportunity table. The columns are: Opp_ID, Opp_Title, No_of_Positions, Specific_Requirements..., Min_CPI_req, No_Active_Backlo..., Student_year_r..., Program_req, Job_Description..., Posted_on.
- Action Output:** Shows the history of actions taken on the database, including insertions, deletions, and selects.
- Status Bar:** Shows "Query Completed".

Opp_ID	Opp_Title	No_of_Positions	Specific_Requirements...	Min_CPI_req	No_Active_Backlo...	Student_year_r...	Program_req	Job_Description...	Posted_on
1	Data Analyst	5	req_file_15.pdf	8.3	FALSE	2	Graphic	job_desc_9.pdf	2024-01-26 16:31:53.125
2	Financial Analyst	1	req_file_10.pdf	8.2	TRUE	3	BA	job_desc_8.pdf	2024-01-18 16:31:53.125
3	Graphic Designer	2	req_file_7.pdf	8.1	FALSE	2	Marketing	job_desc_14.pdf	2024-02-06 16:31:53.125
6	UX Designer	2	req_file_13.pdf	7.9	FALSE	3	CS	job_desc_13.pdf	2024-02-11 16:31:53.125
7	Graphic Designer	2	req_file_4.pdf	8.4	FALSE	2	CS	job_desc_9.pdf	2024-01-22 16:31:53.125
8	Web Developer	2	req_file_10.pdf	8.4	TRUE	3	DS	job_desc_4.pdf	2024-02-03 16:31:53.125
9	UI/UX Designer	1	req_file_15.pdf	8.4	TRUE	4	UI/UX	job_desc_2.pdf	2024-01-22 16:31:53.125
10	Systems Analyst	2	req_file_8.pdf	8.5	TRUE	4	UI/UX	job_desc_10.pdf	2024-01-21 16:31:53.125
11	Software Tester	5	req_file_8.pdf	8.4	TRUE	2	UI/UX	job_desc_5.pdf	2024-02-02 16:31:53.125
12	Data Analyst	1	req_file_1.pdf	8.5	FALSE	2	CS	job_desc_13.pdf	2024-02-03 16:31:53.125
13	Graphic Designer	4	req_file_11.pdf	8.3	FALSE	3	UI/UX	job_desc_1.pdf	2024-01-24 16:31:53.125
14	UX Designer	2	req_file_7.pdf	7.8	FALSE	2	IT	job_desc_3.pdf	2024-02-10 16:31:53.125
15	Software Engineer	4	req_file_9.pdf	8.4	TRUE	4	CS	job_desc_13.pdf	2024-02-12 16:31:53.125
16	Marketing Analyst	3	req_file_6.pdf	8.4	FALSE	2	SE	job_desc_4.pdf	2024-01-22 16:31:53.125
17	Web Developer	3	req_file_4.pdf	8.2	FALSE	2	SE	job_desc_5.pdf	2024-02-06 16:31:53.125
18	UX Designer	3	req_file_16.pdf	7.9	FALSE	2	CS	job_desc_12.pdf	2024-01-21 16:31:53.125
19	Software Tester	2	req_file_14.pdf	8.4	FALSE	2	UI/UX	job_desc_11.pdf	2024-01-21 16:31:53.125

UPDATE

```
1 • Update opportunity set Min_CPI_req = 0.5 + Min_CPI_req;
2
3 • select * from opportunity
```

X 29 01:56:17 Update opportunity set... Error Code: 1142. UPDATE command denied to user 'user1'@'localhost' for table 'opportunity'

DELETION

```
1 • use Assignment2_Database;
2
3 • DELETE FROM Opportunity
4 WHERE Job_Description_file = 'job_desc_15.pdf';
5
6 • select * from opportunity
```



VIEW1

SELECT

A screenshot of MySQL Workbench showing a successful SELECT query on view1. The results grid displays data from the view, and the status bar at the bottom shows a successful query execution.

UPDATION

```
1 •  Update view1 set Min_CPI_req = 0.5 + Min_CPI_req;  
2
```

30 01:57:03 Update view1 set Min_C... Error Code: 1142. UPDATE command denied to user 'user1'@'localhost' for table 'view1'

DELETION

```
1 •  use Assignment2_Database;  
2  
3 •  DELETE FROM view1  
4      WHERE Placement_Method = 'Direct Application';  
5      |  
6 •  select * from view1
```

32 01:57:22 DELETE FROM view1 WH... Error Code: 1142. DELETE command denied to user 'user1'@'localhost' for table 'view1'

2. Mention the situation that violates the referential integrity, show the updates in the table, and how we can solve such problems

There are four main ways in which we can check the violations of referential integrity in DATABASEs, and it mostly has to do with the Foreign Key, which makes reference to the primary keys of the other tables.

- 1) **INSERTION:** If Table A contains the foreign key referencing the primary key of Table B, and we add a new record to Table A with a specified value for the foreign key (let's call it 'x'), an error will occur if 'x' does not correspond to any primary key value in Table B. This error indicates that the referenced value does not exist.

ERROR

Here, we are trying to insert a new value in the table "Student_Academic_Relationship", which has a foreign key (Student_ID for ID of Student dataset), If the new Tuple does not have an already existing ID, then we get an error.

The screenshot shows the MySQL Workbench interface with the following details:

- Left Panel (Schemas):** Shows the database structure. The **Assignment2_Database** is selected, containing the **Tables** table, **Student_Academic_R...** table, and its corresponding **Foreign Keys**.
- Center Panel (Query Editor):** Displays the following SQL code:

```
2 -- select * from student
3
4 • use Assignment2_Database;
5 • select * from student;
6
7 • INSERT INTO Student_Academic_Relationship (Student_ID, Academic_Info_ID)
8 VALUES (00, 01);
9
10 • select * from Student_Academic_Relationship;
```

The line `VALUES (00, 01);` is highlighted with a yellow box.
- Result Grid:** Shows the data for the **Student_Academic_Relationship** table:

Student_ID	Academic_Info_ID
9459	12
7357	14
8898	31
2994	37
2477	39
3658	63
7504	67
4730	78
2529	80
7287	121
8139	131
- Action Output:** Shows the execution log:

Action	Time	Response	Duration / Fetch Time
select * from Student_Academic_Relationship	46 14:54:56	998 row(s) returned	0.0017 sec / 0.00020...
use Assignment2_Database	47 14:57:51	0 row(s) affected	0.0010 sec
select * from student.LIMIT 0,1...	48 14:57:51	957 row(s) returned	0.0018 sec / 0.0031 s...
select * from Student_Academic_Relationship	49 14:57:51	998 row(s) returned	0.0016 sec / 0.00014...
INSERT INTO Student_Academic_Relationship (Student_ID, Academic_Info_ID)	50 14:57:51	Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails ('assignment2_database'...)	0.0048 sec
- Status Bar:** Shows "Query interrupted".

SOLUTION

This problem, does not have a particular solution, as the user must check if the ID already exists, before applying it.

- 2) **MODIFICATION:** If Table A includes a foreign key that references the primary key of Table B, and we update a record (along with its foreign key value) in Table A with a new value for the foreign key (let's call it 'x'), an error will occur if 'x' does not correspond to any primary key value in Table B. This error indicates the absence of the referenced value.

ERROR

Here, we are trying to modify a value in the table "Student_Academic_Relationship", which has a foreign key (Student_ID for ID of Student dataset), but as the data is random, the incremented Value of Student_ID is not in the student table's ID attribute, we get an error.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the schema structure of the 'Assignment2_Database'. The central pane shows a SQL editor with the following code:

```
1 use Assignment2_Database;
2 -- select * from student;
3 select * from Student_Academic_Relationship;
4
5 UPDATE Student_Academic_Relationship SET Academic_Info_ID = Academic_Info_ID + 1;
```

The 'Result Grid' pane below shows the data from the 'Student_Academic_Relationship' table:

Student_ID	Academic_Info_ID
9459	12
7357	14
8898	31
2994	37
2477	39
3658	63
7504	67
4730	78
2529	80
7287	121
8139	131
8167	139
8200	144

The 'Action Output' pane at the bottom shows the execution log:

Action	Time	Response	Duration / Fetch Time
select * from student LIMIT 0, 1...	15:00:45	957 row(s) returned	0.0011 sec / 0.0028 s...
UPDATE Student_Academic_Re...	15:00:45	Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails ('assignment2_database...' 0 row(s) affected)	0.0078 sec
use Assignment2_Database	15:01:26	0 row(s) affected	0.00047 sec
select * from Student_Academi...	15:01:26	998 row(s) returned	0.0013 sec / 0.00020...
UPDATE Student_Academic_Re...	15:01:26	Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails ('assignment2_database...' 0 row(s) affected)	0.0038 sec

A yellow oval highlights the error message in the log: "Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails ('assignment2_database...' 0 row(s) affected)".

SOLUTION

This problem, does not have a particular solution, as the user must check if the modified ID already exists, before changing it.

3) **DELETION:** If Table A includes a foreign key that references the primary key of Table B, and we delete a specific row solely from Table B (without affecting Table A), and if that row's primary key is referenced elsewhere in Table A, an error will occur indicating that the parent row's value cannot be deleted because it is still referenced by Table A. Deleting such a row from Table B without removing the corresponding referencing rows from Table A would result in Table A referencing a non-existent value from Table B, thereby violating referential integrity

ERROR

Here, we are trying to delete the columns (tuples) of the students, whose Student ID is less than 20. This will give an error as the Table “Student Academic Relationship” is also using it.

The screenshot shows the MySQL Workbench interface. In the top-left, the schema is set to 'Assignment2_Database'. The main area contains a query editor with the following SQL code:

```

1 • use Assignment2_Database;
2 • select * from student;
3 • --- select * from Student_Academic_Relationship;
4
5 • Delete From Student where Student_ID < 20
6

```

The 'Delete From Student where Student_ID < 20' line is highlighted with a yellow box. Below the editor is a 'Result Grid' showing student data. At the bottom, the 'Action Output' pane displays the execution log:

Action	Time	Response	Duration / Fetch Time
select * from Student_Academi...	15:06:57	998 row(s) returned	0.0052 sec / 0.00014...
Delete From Student where Stu...	15:06:57	Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails ('assignment2_database'.'student', CONSTRAINT 'fk_academic_relationship' FOREIGN KEY ('Student_ID') REFERENCES 'student' ('Student_ID'))	0.0067 sec
use Assignment2_Database	15:07:14	0 row(s) affected	0.00032 sec
select * from student LIMIT 0, 1...	15:07:14	957 row(s) returned	0.0010 sec / 0.0030...
Delete From Student where Stu...	15:07:14	Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails ('assignment2_database'.'student', CONSTRAINT 'fk_academic_relationship' FOREIGN KEY ('Student_ID') REFERENCES 'student' ('Student_ID'))	0.0020 sec

A yellow oval highlights the error message in the log.

SOLUTION

We can address these concerns by implementing referential actions such as ON DELETE CASCADE and ON UPDATE CASCADE. Essentially, if a record is deleted or updated in Table B, all corresponding records in Table A will be automatically deleted or updated accordingly. This mechanism guarantees the maintenance of referential integrity, preventing any violations.

```

    79      );
80
81 • CREATE TABLE Student_Academic_Relationship (
82     Student_ID INT,
83     Academic_Info_ID INT,
84     PRIMARY KEY (Student_ID, Academic_Info_ID),
85     FOREIGN KEY (Student_ID) REFERENCES Student(Student_ID)
86     ON DELETE CASCADE
87     ON UPDATE CASCADE,
88     FOREIGN KEY (Academic_Info_ID) REFERENCES Academic_Info(Academic_Info_ID)
89     ON DELETE CASCADE
90     ON UPDATE CASCADE,
91   );
92
93 • CREATE TABLE Placement_Student (
94     Placement_ID INT,
95     Student_ID INT,
96     CONSTRAINT fk_placement FOREIGN KEY (Placement_ID) REFERENCES Placement(Placement_ID)

```

After applying Cascading, the Delete command worked well. You can see, that after the update, the columns with student ID 5, 15, 16 (<20) have been deleted, and in the left corner, it shows that the Foreign key has CASCADE applied.

BEFORE CASCADING

```

1 • use Assignment2_Database;
2 • select * from student;
3 -- select * from Student_Academic_Relationship;
4
5 • Delete From Student where Student_ID < 20
6

```

Student_ID	Student_First_Na...	Student_Middle_Na...	Student_Last_Na...	Department	Active_Backlog	Gender	Year	Student_Image	Minors	Student_Email_Id
5	Sarah	Jacob	Davis	History	4	Other	2	student_image_5.jpg	Psychology	sarah.davis@itgn.ac.i
15	Alice	HULL	Brown	Biology	2	Male	3	student_image_15.jpg	Physics	alice.brown@itgn.ac.i
16	Sarah	Elizabeth	Johnson	Physics	0	Female	3	student_image_16.jpg	NULL	sarah.johnson@itgn.e
50	Michael	Olivia	Smith	Psychology	4	Female	1	student_image_50.jpg	Chemistry	michael.smith@itgn.a
60	Jane	William	Brown	Sociology	0	Other	3	student_image_60.jpg	Art	jane.brown@itgn.ac.i
61	David	HULL	Davis	Art	1	Female	1	student_image_61.jpg	History	david.davis@itgn.ac.i
62	Alice	HULL	Davis	Economics	4	Male	4	student_image_62.jpg	Physics	alice.davis@itgn.ac.ir
64	Bob	Olivia	Johnson	Art	0	Male	1	student_image_64.jpg	History	bob.johnson@itgn.ac
93	Michael	HULL	Davis	Art	3	Other	3	student_image_93.jpg	Chemistry	michael.davis@itgn.a
107	Emily	HULL	Johnson	Chemistry	4	Female	3	student_image_107.jpg	English	emily.johnson@itgn.a
119	David	Elizabeth	Lee	Sociology	3	Other	4	student_image_119.jpg	Art	david.lee@itgn.ac.in
182	Emily	William	Brown	English	0	Other	2	student_image_182.jpg	Economics	emily.brown@itgn.ac
104	Michael	Elizabeth	Smith	English	4	Other	4	student_image_104.jpg	Art	michael.smith@itgn.a

Action Output: student 6

Time	Action	Response	Duration / Fetch Time
64 15:06:57	select * from Student_Academi...	998 row(s) returned	0.0052 sec / 0.00014...
65 15:06:57	Delete From Student where Stu...	Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails ('assignment2_database'."student".'Student_ID' = 'placement_student'."Student_ID")	0.0067 sec
66 15:07:14	use Assignment2_Database	0 row(s) affected	0.00032 sec
67 15:07:14	select * from student LIMIT 0, 1...	957 row(s) returned	0.0010 sec / 0.0030...
68 15:07:14	Delete From Student where Stu...	Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails ('assignment2_database'."student".'Student_ID' = 'placement_student'."Student_ID")	0.0020 sec

AFTER CASCADING

```

1 • use Assignment2_Database;
2 -- select * from student;
3
4 • Delete From Student where Student_ID < 20;
5
6 • select * from student;

```

Student_ID	Student_First_Na...	Student_Middle_Na...	Student_Last_Na...	Department	Active_Backlog	Gender	Year	Student_Image	Minors	Student...
50	Michael	Olivia	Smith	Psychology	4	Female	1	student_image_50.jpg	Chemistry	michael.s...
60	Jane	William	Brown	Sociology	0	Other	3	student_image_60.jpg	Art	jane.brow...
61	David	NULL	Davis	Art	1	Female	1	student_image_61.jpg	History	david.dav...
62	Alice	NULL	Davis	Economics	4	Male	4	student_image_62.jpg	Physics	alice.davi...
64	Bob	Olivia	Johnson	Art	0	Male	1	student_image_64.jpg	History	bob.johns...
93	Michael	NULL	Davis	Art	3	Other	3	student_image_93.jpg	Chemistry	michael.c...
107	Emily	NULL	Johnson	Chemistry	4	Female	3	student_image_107.jpg	English	emily.joh...
119	David	Elizabeth	Lee	Sociology	3	Other	4	student_image_119.jpg	Art	david.lee...
182	Emily	William	Brown	English	0	Other	2	student_image_182.jpg	Economics	emily.brow...
194	Michael	Elizabeth	Smith	English	4	Other	4	student_image_194.jpg	Art	michael.s...
206	John	NULL	Doe	Art	2	Other	4	student_image_206.jpg	Psychology	john.doe...

student 3

Action Output

T...	Ac...	Response
38	15...	us... 0 row(s) affected
39	15...	sel... 957 row(s) returned
40	15...	us... 0 row(s) affected
41	15...	Del... 3 row(s) affected
42	15...	sel... 954 row(s) returned

Query Completed

4) **UPDATION:** If Table A contains a foreign key referencing the primary key of Table B, and we update a specific row solely within Table B (without affecting Table A), where the primary key is referenced elsewhere in Table A, an error will occur indicating that the parent row's value cannot be updated because it is still referenced by Table A with its original value. Updating such a row in Table B without updating the corresponding referencing rows in Table A would result in Table A referencing an outdated value from Table B, thereby violating referential integrity

ERROR

Here, we are trying to UPDATE the columns (tuples) of the students,to increase the ACTIVE backlog to +1. This will give an error as the Table "Student Academic Relationship" is also using it.

root - Warning - not supported

```

use Assignment2_Database;
-- select * from Student_Academic_Relationship;
select * from student;

UPDATE Student set Active_Backlog = Active_Backlog +1;
select * from student;

```

Result Grid

Student_ID	Student_First_Na...	Student_Middle_Na...	Student_Last_Na...	Department	Active_Backlog	Gender	Year	Student_Image	Minors	Stud...
50	Michael	Olivia	Smith	Psychology	4	Female	1	student_image_50.jpg	Chemistry	micha...
60	Jane	William	Brown	Sociology	0	Other	3	student_image_60.jpg	Art	jane...
61	David	NULL	Davis	Art	1	Female	1	student_image_61.jpg	History	david...
62	Alice	NULL	Davis	Economics	4	Male	4	student_image_62.jpg	Physics	alice...
64	Bob	Olivia	Johnson	Art	0	Male	1	student_image_64.jpg	History	bob...
93	Michael	NULL	Davis	Art	3	Other	3	student_image_93.jpg	Chemistry	micha...
107	Emily	NULL	Johnson	Chemistry	4	Female	3	student_image_107.jpg	English	emily...
119	David	Elizabeth	Lee	Sociology	3	Other	4	student_image_119.jpg	Art	david...
182	Emily	William	Brown	English	0	Other	2	student_image_182.jpg	Economics	emily...
194	Michael	Elizabeth	Smith	English	4	Other	4	student_image_194.jpg	Art	micha...
206	John	NULL	Doe	Art	2	Other	4	student_image_206.jpg	Psychology	john...

Action Output

Time	Action	Response
16:32:56	UPDATE Student set Active_Backlog = A...	954 row(s) affected Rows matched: 954 Changed: 954 Warnings: 0
16:32:56	select * from student LIMIT 0, 1000	954 row(s) returned
16:33:49	use Assignment2_Database	0 row(s) affected
16:33:49	select * from student LIMIT 0, 1000	954 row(s) returned
16:33:49	UPDATE Student set Student_ID = Stud...	Error Code: 1062. Duplicate entry '61' for key 'student.PRIMARY'

SOLUTION

We can address these concerns by implementing referential actions such as ON DELETE CASCADE and ON UPDATE CASCADE. Essentially, if a record is deleted or updated in Table B, all corresponding records in Table A will be automatically deleted or updated accordingly. This mechanism guarantees the maintenance of referential integrity, preventing any violations.

BEFORE CASCADING

You can look at the figure above, the backlogs read (4, 0, 1, 4, 0, 3)

AFTER CASCADING

Now, the backlogs have incremented by 1 to (5, 1, 2, 5, 1, 4)

The screenshot shows the MySQL Workbench interface. On the left, the Schemas tree displays various database objects like Opportunity, Person_of_Contact, Placement, Placement_Student, Student, and Student_Academic_Relationship. A yellow circle highlights the 'Foreign Keys' section under Student_Academic_Relationship, which lists 'student_academic_ibfk_1' and 'student_academic_ibfk_2'. Below this, 'Object Info' and 'Session' tabs are visible. A yellow box highlights the 'Definition:' section for 'student_academic_ibfk_1', showing 'Target' as 'Student (Student_ID → Student_ID)', 'On Update' as 'CASCADE', and 'On Delete' as 'CASCADE'. The main area contains a SQL editor with the following code:

```

1 • use Assignment2_Database;
2   -- select * from Student_Academic_Relationship;
3
4 • UPDATE Student set Active_Backlog = Active_Backlog +1;
5 • select * from student;
6

```

The results grid shows a table with columns: Student_ID, Student_First_Na..., Student_Middle_Na..., Student_Last_Na..., Department, Active_Backlog, Gender, Year, Student_Image, Minor. The data includes rows for Michael, Jane, David, Alice, Bob, Michael, Emily, David, Emily, William, Michael, and John. The action output shows the following log entries:

T...	A...	Response
43	15...	0 row(s) affected
44	15...	954 row(s) returned
45	15...	0 row(s) affected
46	15...	UP... 954 row(s) affected Rows matched: 954 Changed: 954 Warnings: 0
47	15...	sel... 954 row(s) returned

At the bottom, a message says 'Query Completed'.

Note: In doing this question, we were getting repeated errors, even after CASCADING, and this was due to some other Tables, also having Foreign Keys (Like Application), hence we had to CASCADE them also.

The screenshot shows the MySQL Workbench interface with a query editor containing SQL code. A yellow circle highlights the 'CONSTRAINT fk_student FOREIGN KEY (Student_ID) REFERENCES Student(Student_ID)' clause. Another yellow circle highlights the 'CONSTRAINT fk_student_app FOREIGN KEY (Student_ID) REFERENCES Student(Student_ID)' clause. The code includes several CREATE TABLE statements and constraints:

```

99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120

```

The Action Output section at the bottom is empty.

3.3 Responsibility of G1 & G2:

Using the optimized ER diagram, write five operations, their corresponding nested SQL queries, and tuple relational calculus/relational algebra with the following specifications:

1. Two queries must throw an error due to a violation of constraints specified by G1.
2. One of the functions should involve the storage of an image along with a caption in the database.
3. Include cases of natural join, outer join, renaming, two or more different kinds of aggregate functions, and case statements in one or more queries.

Clearly mention the specifications each operation satisfies. For, e.g., if operation f1 involves storing an image, write "f1 satisfies specification 2". Also, submit screenshots of the results.

1. Operation: Here, we are attempting to use the UNION operator to combine the results of two SELECT statements. The first SELECT statement selects columns from the student table, while the second SELECT statement selects columns from the Academic_info table

Relational Algebra:

$$\sigma_{\text{dept}=\text{"CSE"}}(\text{student}) \cup \sigma_{\text{SSAC_or_Not}=\text{"YES}}(\text{Academic_Info})$$

SQL Query:

```
SELECT * FROM student WHERE Department = 'CSE'  
UNION  
SELECT * FROM Academic_Info WHERE SSAC_or_Not = 'YES';
```

Action	Output	Time	Action	Response	Duration / Fetch Time
✗ 180 12:33:11	SELECT * FROM student WHERE Department = 'CSE' UNION SELECT * FROM Academic_Info WHERE SSAC_or_Not = 'YES'			Error Code: 1222. The used SELECT statements have a different number of columns	0.0038 sec

'This Operation satisfies specification 1'

This operation gives an error due to violating constraints specified by G1.

Reasoning: The two SELECT statements have different column names and data types, which will result in a type mismatch error when the UNION operator tries to combine the two result sets. Specifically, the student table columns are Student_id, Student_First_Name, Student_Middle_Name, Student_Last_Name, dept, Department, Active_Backlog, Gender, Year, Student_Image, Minor, Student_Email_Id, Contact_number.

The Academic_Info table columns are Academic_Id, Transcript_File, CPI, SSAC_or_Not. The two tables have different column names and column numbers.

2. Operation: Inserting an entry into the Student table with having an invalid year.

Relational Algebra/Calculus:

```
Student ← Student ∪ {21110012, 'Adit', 'Bhai', 'Kaushik', 'CSE', 0, 'Male', 7,  
'student_image_5.jpg', 'Business', 'adit@iitgn.ac.in', 1234567891}
```

SQL Query: INSERT INTO Student VALUES (21110012, 'Adit', 'Bhai', 'Kaushik', 'CSE', 0, 'Male', 7, 'student_image_5.jpg', 'Business', 'adit@iitgn.ac.in', 1234567891)

Error Code: 3819. Check constraint 'chk_year' is violated.

A screenshot of a database query log window. The log shows a single failed query with the following details:
- Row ID: 189
- Time: 12:56:08
- Statement: INSERT INTO Student VALUES (21110012, 'Adit', 'Bhai', 'Kaushik', 'CSE', 0, 'Male', ...
- Error: Error Code: 3819. Check constraint 'chk_year' is violated.
- Duration: 0.0083 sec

'This Operation satisfies specification 1'

Reasoning: We have set the constraint such that Student.year can only take values from set {1, 2, 3, 4}. But in the above SQL Query we are trying to insert Student with year 7, which is violating the constraint.

3. Operation: Here we are performing join over placement and opportunity, on the column Job_designation of placement and Opp_title in Opportunity.

SQL Query:

```
SELECT  
    Placement.Placement_ID AS ID,  
    Placement.Placement_Method AS Method,  
    Placement.Company_Name AS Company,  
    Placement.Job_Designation AS Designation,  
    Opportunity.Opp_Title AS Opportunity_Title,  
    Opportunity.Min_CPI_req AS Minimum_CPI_Requirement  
FROM
```

Placement

LEFT OUTER JOIN

Opportunity ON Opportunity.Opp_Title = Placement.Job.Designation;

Relational Algebra/Calculus:

$\pi_{\text{Placement_ID} \rightarrow \text{ID}, \text{Placement_Method} \rightarrow \text{Method}, \text{Company_Name} \rightarrow \text{Company}, \text{Job_Designation} \rightarrow \text{Designation}, \text{Opp_Title} \rightarrow \text{Opp_Title}}$

$\text{Opportunity_Title}, \text{Min_CPI_req} \rightarrow \text{Minimum_CPI_Requirement}$

$(\sigma_{\text{Opportunity.Opp_Title} = \text{Placement.Job_Designation}} (\text{Placement} \bowtie_{\text{Job_Designation}=\text{Opp_Title}} \text{Opportunity}))$

ID	Method	Company	Designation	Opportunity_Title	Minimum_CPI.Requireme...
10	Campus Placement	Tesla	Systems Analyst	Systems Analyst	7.4
10	Campus Placement	Tesla	Systems Analyst	Systems Analyst	7.2
10	Campus Placement	Tesla	Systems Analyst	Systems Analyst	7.2
10	Campus Placement	Tesla	Systems Analyst	Systems Analyst	7.5
20	Direct Application	Finance Gurus	Software Engineer	Software Engineer	7.6
20	Direct Application	Finance Gurus	Software Engineer	Software Engineer	7.3
20	Direct Application	Finance Gurus	Software Engineer	Software Engineer	7.0
20	Direct Application	Finance Gurus	Software Engineer	Software Engineer	7.4
42	Campus Placement	Biz Geniuses	Graphic Designer	Graphic Designer	7.3
42	Campus Placement	Biz Geniuses	Graphic Designer	Graphic Designer	7.5
42	Campus Placement	Biz Geniuses	Graphic Designer	Graphic Designer	7.4
42	Campus Placement	Biz Geniuses	Graphic Designer	Graphic Designer	7.3
42	Campus Placement	Biz Geniuses	Graphic Designer	Graphic Designer	7.4
42	Campus Placement	Biz Geniuses	Graphic Designer	Graphic Designer	7.1
96	Direct Application	IBM	Software Tester	Software Tester	7.3
96	Direct Application	IBM	Software Tester	Software Tester	7.4
96	Direct Application	IBM	Software Tester	Software Tester	7.2
96	Direct Application	IBM	Software Tester	Software Tester	7.3
96	Direct Application	IBM	Software Tester	Software Tester	7.4
96	Direct Application	IBM	Software Tester	Software Tester	7.4
96	Direct Application	IBM	Software Tester	Software Tester	7.1
154	Direct Application	Samsung	Marketing Analyst	Marketing Analyst	7.6
154	Direct Application	Samsung	Marketing Analyst	Marketing Analyst	7.6

'This Operation satisfies specification 3'

Reasoning: We are performing join over placement and opportunity, on the column Job_designation of placement and Opp_title in Opportunity. We are also renaming all of the columns in this SQL query.

4. Operation: We are categorizing the salary into three categories in Placement using case statement. If salary is ≥ 95000 , then it is High and if ≤ 70000 then it is low else medium

Relational Algebra/Calculus:

$\pi_{\text{Company_Name}, \text{Job_Designation}, \text{Salary}}$,

$\text{Salary_Category}(\sigma_{\text{Salary} \geq 95000 \vee (\text{Salary} \leq 70000) \vee (\text{Salary} \geq 70000 \wedge \text{Salary} \leq 95000)}(\text{Placement}))$

SQL Query:

```

SELECT
    Placement.Company_Name,
    Placement.Job_Designation,
    Placement.Salary,
    CASE
        WHEN Placement.Salary >= 95000 THEN 'High'
        WHEN Placement.Salary <= 70000 THEN 'Low'
        ELSE 'Medium'
    END AS Salary_Category
FROM Placement;

```

Company_Name	Job_Designation	Salary	Salary_Category
Tesla	Systems Analyst	117244	High
Finance Gurus	Software Engineer	62849	Low
Biz Geniuses	Graphic Designer	90496	Medium
IBM	Software Tester	80156	Medium
Samsung	Marketing Analyst	67288	Low
Adobe	Software Engineer	118261	High
Code Crafters	Business Analyst	68305	Low
Network Pros	Web Developer	90740	Medium
Google	Business Analyst	69565	Low
Oracle	Software Engineer	64304	Low
Tesla	Data Analyst	106758	High
Code Innovate	Graphic Designer	68384	Low
Biz Geniuses	Network Engineer	100069	High
Samsung	Network Engineer	71964	Medium
Facebook	Systems Analyst	98386	High
Amazon	Data Analyst	75255	Medium
Microsoft	Web Developer	61156	Low
Soft Creations	Systems Analyst	115424	High
Data Insights	Graphic Designer	103596	High

'This Operation satisfies specification 3'

Reasoning: Here we are making use of Case statement. We are categorizing the salary into three categories in Placement using Case statement. If salary is ≥ 95000 , then it is High, and if ≤ 70000 then it is Low, else Medium

5. Operation: We are inserting image into “student” table. Then we are adding additional attribute “student_image-caption” into the table and inserting image and caption for the image for student with student_id = 10.

Relational Algebra/Calculus:

Student \leftarrow Student $\cup \{ (10, \text{LOAD_FILE('bhotwikk.jpg')}) \}$

SQL Query:

```
UPDATE student SET Student_Image = LOAD_FILE('bhotwikk.jpg')
WHERE Student_ID = 10;
```

```
mysql> INSERT INTO student (Student_ID, Student_Image) VALUES (10, LOAD_FILE('/Users/aditkaushik/dbms/assignment2/data/bhotwikk.jpg'));
ERROR 1364 (HY000): Field 'Student_First_Name' doesn't have a default value
mysql> UPDATE student
-> SET Student_Image = LOAD_FILE('/Users/aditkaushik/dbms/assignment2/data/bhotwikk.jpg')
-> WHERE Student_ID = 10;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0
mysql> select * from student where student_id = 10
-> ;
```

```
ALTER TABLE student ADD COLUMN Student_Image_Caption varchar(100);
UPDATE student SET Student_Image_Caption = "student_image"
WHERE Student_ID = 10;
```

$\text{Student} \leftarrow \text{Student} \cup \{ \text{student_image} \}$

```
mysql> UPDATE student SET Student_Image_Caption = "student_image" WHERE Student_ID = 10;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select student_image_caption from student where student_id = 10;
+-----+
| student_image_caption |
+-----+
| student_image          |
+-----+
1 row in set (0.05 sec)
```

'This Operation satisfies specification 2'

Important Links

- 5) ER Diagram: Draw.io file:

<https://app.diagrams.net/?client=1#G1yCpFEy0jyzqNLsKFg6Lim8za2PPASyPd%7B%22pageId%22%3A%22n4RG28VcgylvL5a97sLb%22%7D>

- 6) Code for the Data Generation process: Google Colab notebook:

<https://colab.research.google.com/drive/1thD8FtA0pBjFwbtnWxz85L4WhKXOa4s->

- 7) Generated Dataset for the Database: Google Drive:

https://drive.google.com/drive/folders/1lr40jKZI_w58t7_DY7AVztuXKtXFc2A2

- 8) SQL Files, and data dump:

https://drive.google.com/drive/folders/1_MqzNJfXfNdrbH-PaVhR7fRd7r1Yt9vL

Contributions:

Name	Group	Roll Number	Contribution
Anugu Arun Reddy	G1	21110029	<ul style="list-style-type: none">• Contributed to generating data.• Brainstormed The working of indexing and user-defined properties.• Contributed to debugging the code.
Aditya Deshmukh	G1	21110014	<ul style="list-style-type: none">• Created a data generation script to streamline processes.• Conducted a thorough analysis of errors in the previous assignment and implemented necessary revisions.• Played a key role in extending column and table indexing functionality.
Abhay Kumar Upparwal	G1	21110004	<ul style="list-style-type: none">• Contributed in generating the data of all the tables following the constraint and relations between them.• Contributed in writing operation for part 3.3.
Aaryan Darad	G1	21110001	<ul style="list-style-type: none">• Contributed to Generation of Data• Contributed to the indexing of columns and table extension.• Formatted the document and corrected missing points or errors.
Gaurav Joshi	G2	21110065	<ul style="list-style-type: none">• Worked on G2 Question, making the user, giving permission and making queries.• Brainstorming changes based on the suggestions of the TA• Remaking the ER Diagram in Draw.io• Working on the Referential Integrity problem, making the Cascading updates and Getting snapshots of the Queries and SQL

			files.
Adit Kaushik	G2	21110010	<ul style="list-style-type: none"> • Wrote down the nested SQL queries for the operations which included queries such as inserting images along with captions to the database and case statements. • Contributed & checked relational algebra for all the operations performed in Q 3.3
More Rutwik	G2	21110133	<ul style="list-style-type: none"> • Listed down the operations that throw errors due to violation of constraints specified on attributes. • Wrote down the nested SQL queries for the operations which included queries such as joins, renaming, and using case statements. • Wrote relational algebra for all the operations performed in Q 3.3 • Worked on Q 3.3, making the user, giving permission, and making queries.
Ahaan Giriya	G2	21110015	<ul style="list-style-type: none"> • Contributed in indexing of user-defined data type • Helped in Question 3.3 • Formatted the document