



## SCHOOL MANAGEMENT SYSTEM

by

Rudhar Sharma – C0871405

Reea Sagar – C0869344

Fahad Ismail Mohammed – C0872742

Akhilesh K Saju – C0879963

Praveen Pallipurath – C0879905

## TABLE OF CONTENTS

• Introduction .....	3
• Product/Service/Methodology .....	4-9
• Visual Data .....	10
• Normalization.....	11
• SQL queries.....	12-20
• Normalized tables.....	21-29
• Physical Database.....	30-45
• Conclusion .....	46
• Key Takeaways .....	47



## INTRODUCTION TO THE PROCESS:

- The purpose of this project is to demonstrate a student database management system in its simplest form that uses a digital tracking system to maintain a record of all the students. However, in practicality, the system has much more complex functions.
- It maintains a record of every student-related data ranging from fees and financial records, examination records, transport facilities being provided by the school and availed by the students along with the usage of libraries, laboratory, computer lab and other school facilities used regularly by the students.
- A student database management system allows schools to save and access these records as needed by them, thus simplifying the work of the school administration team.
- The attendance management system uses a biometric system or an access card to maintain a quick and accurate record of the student and school staff attendance. The system saves class teachers tedious minutes every morning calling out the class attendance and provides with the administration team on timely summaries and records of regular latecomers, absentees etc. when needed.
- The system also allows for attendance updates to be sent to parents. It enables schools to track the attendance of the teachers and the various staff and calculate leaves and working days with ease and accuracy.

## ENTITIES AND ATTRIBUTES

An entity in DBMS (Database management System) is **a real-world thing or a real-world object which is distinguishable from other objects in the real world**. For example, a car is an entity.

The entities and attributes we used in our project:

### Student:

**Attributes: AddressID, First\_name, Last\_name, parent\_name, Dob, Mobile, Email, Joining\_date.**

**Primary Key for student is Student\_ID.**

**Foreign Key for student is Address\_ID.**

student	
PK	<u>student_ID</u>
FK	address_id
	first_name
	last_name
	guardian_name
	dob
	mobile
	email
	joining_date

## Staff:

Attributes: Staff\_Id, First\_name, Last\_name, Phone, Email, Dob, Designation.

Primary key is Staff\_ID.

Staff	
PK	<u>staff_ID</u>
	first_name
	last_name
	phone
	email
	dob
	designation

## Courses:

Attributes: Course\_ID, Grade\_ID, name, Description, Credits.

Primary key for Courses is Course\_ID.

Foreign Key for Courses is Grade\_ID.

course	
PK	<u>course_id</u>
FK	grade_id
	name
	description
	credits

### Class:

Attributes: Class\_ID, Grade\_Id, Staff\_id, Year.

Primary key for Class Is Class\_ID.

Foreign key for Class Is Grade\_ID.

Foreign key for Class Is Staff\_ID.

class	
PK	<u>class_enrolled_id</u>
FK	grade_id
FK	staff_id
	year_of_attempt

### Exam:

Attributes: Exam\_Id, Course\_id, Exam\_type\_id, Student\_id, Start\_date.

Primary Key is Exam\_ID.

Foreign Key is Exam\_type\_ID.

Foreign Key is student\_ID.

Foreign Key is Course\_ID.

exam	
PK	<u>exam_val_id</u>
FK	course_id
FK	exam_type_id
FK	student_id
	start_date

### Exam result:

Attributes: Exam\_id, Course\_id, Marks.

Primary key for Exam result is Exam\_ID.

Foreign key for Exam result is Student\_ID.

Foreign key for Exam result is Course\_ID.

exam_result	
PK	<u>exam_result_id</u>
FK	student_id
FK	course_id
	marks

### Exam Type:

Attributes: Exam\_type\_id, Type, Desc.

Primary key for Exam type is Exam\_type\_id.

exam_type	
PK	<u>exam_type_id</u>
	type
	desc

## Student Council:

Attributes: Student\_council\_ID, Student\_ID, Designation.

Primary key for Student Council is Student\_council\_id.

Foreign key for Student Council is Student\_id.

student_council	
PK	<u>student_council_ID</u>
FK	student_ID
	designation

## Attendance:

Attributes: Attendance\_Id, Student-Id, Date, Status.

Primary key for attendance is attendance\_id.

Foreign key for attendance is Student\_id.

attendance	
PK	<u>attendance_id</u>
FK	student_id
	current_date
	status

## Grades:

Attributes: Grade\_id, Name, Desc.

Primary key for Grades is grade\_id.

grade	
PK	<u>grade_id</u>
	name
	desc

## Library:

**Attributes:** Library\_id, Student\_id, Status.

**Primary key for Library is Library\_id.**

**Foreign key for Library is Student\_id.**

**Foreign key for Library is Book\_id.**

library	
PK	library_id
FK	student_id
FK	book_id
	status

## Book:

**Attributes:** Book\_id, Name, Desc.

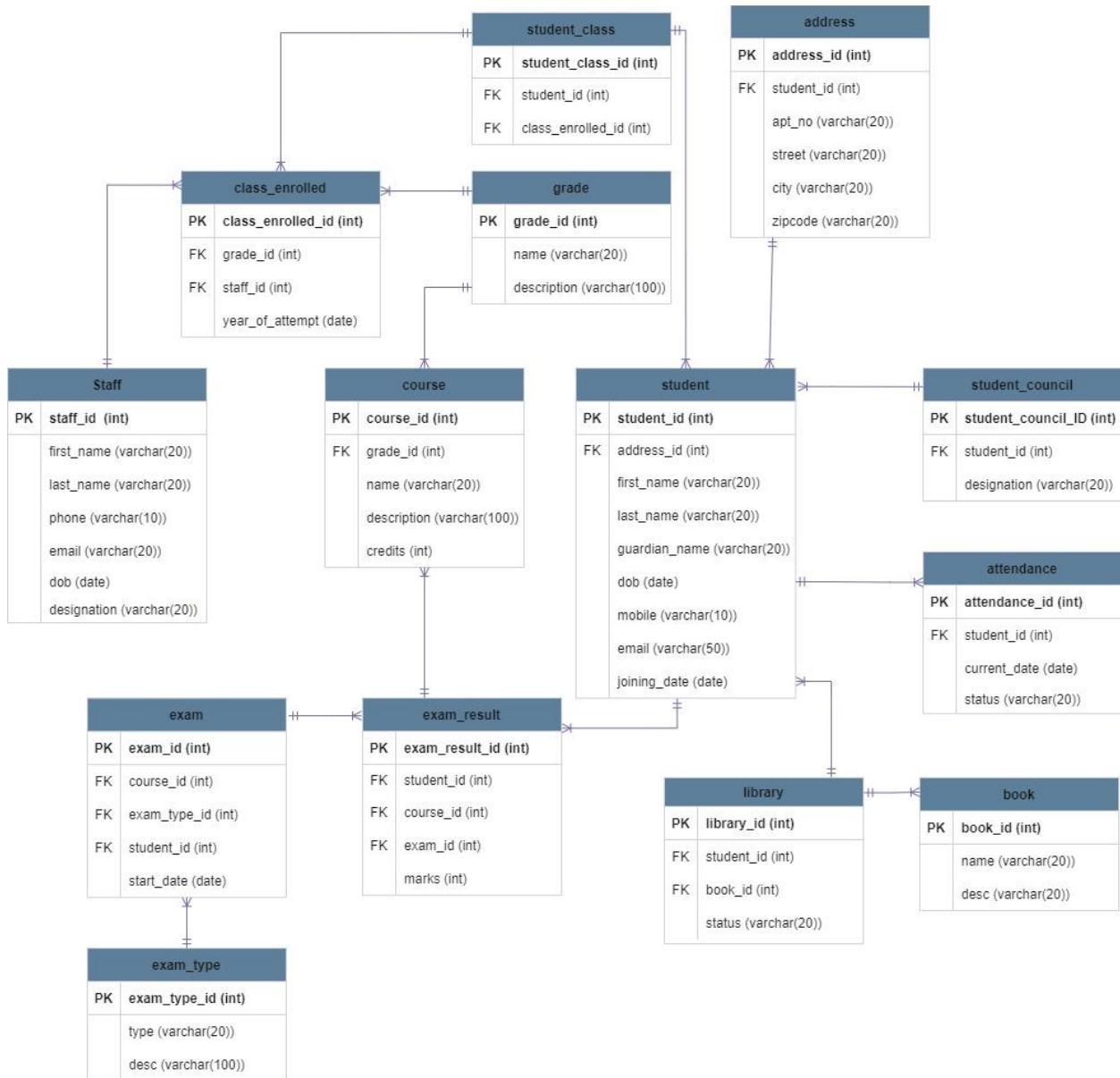
**Primary key for Book is Book\_id.**

book	
PK	book_id
	name
	desc

**Mandatory attributes:** Book\_ID and Name

**Optional Attributes:** Desc

## Entity Relational Diagram:



**2. Normalization:** It is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies. Normalization rules divides larger tables into smaller tables and links them using relationships. The purpose of Normalization in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.

➤ Criteria for 1NF, 2NF and 3NF

- **1NF:**
  - Each table cell should contain a SINGLE VALUE.
  - Each record needs to be UNIQUE
- **2NF:**
  - Rule-1 Be in 1NF.
  - Rule-2 Single Column Primary Key that does not functionally dependent on any subset of candidate key relation.
- **3NF:**
  - Rule-1 Be in 2NF.
  - Rule-2 has no transitive functional dependencies.

❖ In all the tables (*Student, Class, Course, Grade, Student\_Class, Address, Staff, Student\_Council, Attendance, Exam, Exam\_result, Exam\_Type, Library and Book*).

▪ **1NF:**

- ✓ Primary Key is defined.
- ✓ No multivalued Columns and repeating groups found.
- ✓ Hence, they are in 1NF.

▪ **2NF:**

- ✓ No partial dependencies found in all the tables.
- ✓ Hence, they are in 2NF.

▪ **3NF:**

- ✓ No transitive dependencies found in all the tables.
- ✓ Hence, they are in 3NF.

## SQL QUERIES FOR DATA INSERTION AND MODIFICATION

```
SELECT * FROM STUDENT;  
SELECT * FROM ADDRESS;  
SELECT * FROM STUDENT_COUNCIL;  
SELECT * FROM ATTENDANCE;  
SELECT * FROM BOOK;  
SELECT * FROM LIBRARY;  
SELECT * FROM EXAM_TYPE;  
SELECT * FROM STAFF;  
SELECT * FROM GRADE;  
SELECT * FROM STUDENT_CLASS;  
SELECT * FROM CLASS_ENROLLED;  
SELECT * FROM COURSE;  
SELECT * FROM EXAM_RESULT;  
SELECT * FROM EXAM;
```

```
INSERT INTO "STUDENT" (STUDENT_ID, ADDRESS_ID, FIRST_NAME, LAST_NAME, GUARDIAN_NAME,  
DOB, MOBILE, EMAIL, JOINING_DATE)  
VALUES ('1', '1', 'akhilesh', 'saju', 'saju', TO_DATE('1997-12-05 12:09:45', 'YYYY-MM-DD HH24:MI:SS'),  
'6478325376', 'akhileshsaju@gmail.com', TO_DATE('2022-12-06 12:10:17', 'YYYY-MM-DD  
HH24:MI:SS'))
```

```
INSERT INTO "ADDRESS" (  
address_id,  
apt_no,  
street,  
city,  
zipcode  
)
```

```
VALUES (1, '14', 'McLaughlin', 'Brmapton', 'L5X6W1')
```

```
INSERT INTO "STUDENT" (STUDENT_ID, ADDRESS_ID, FIRST_NAME, LAST_NAME, GUARDIAN_NAME,  
DOB, MOBILE, EMAIL, JOINING_DATE)
```

```
VALUES ('2', '2', 'praveen', 'saji', 'saji', TO_DATE('1997-06-03 12:11:02', 'YYYY-MM-DD HH24:MI:SS'),  
'6475687689', 'praveen@hotmail.com', TO_DATE('2022-12-06 15:10:17', 'YYYY-MM-DD HH24:MI:SS'))
```

```
INSERT INTO "ADDRESS" (
```

```
address_id,
```

```
apt_no,
```

```
street,
```

```
city,
```

```
zipcode
```

```
)
```

```
VALUES (2, '69', 'Williams Parkway', 'Brmapton', 'L3X2C1')
```

```
create table Student_Council(
```

```
student_council_id int primary key,
```

```
student_id int,
```

```
Designation varchar(20),
```

```
CONSTRAINT fk_student
```

```
FOREIGN KEY (Student_id)
```

```
REFERENCES student(Student_id)
```

```
)
```

```
INSERT INTO "STUDENT_COUNCIL" (STUDENT_COUNCIL_ID, STUDENT_ID, DESIGNATION) VALUES ('1', '1', 'm')
```

```
INSERT INTO "STUDENT_COUNCIL" (STUDENT_COUNCIL_ID, STUDENT_ID, DESIGNATION) VALUES ('2', '2', 'p')
```

```
create table Attendance(  
Attendance_id int primary key,  
Student_id int,  
Current_Date date not null,  
Status varchar(20) CONSTRAINT check_status_of_attendance  
    CHECK (Status in ('present','absent')) disable,  
CONSTRAINT fk_student_check  
    FOREIGN KEY (Student_id)  
    REFERENCES student(Student_id)  
)
```

```
INSERT INTO "ATTENDANCE" (ATTENDANCE_ID, STUDENT_ID, CURRENT_DATE, STATUS) VALUES ('1', '2', TO_DATE('2022-12-06 12:19:02', 'YYYY-MM-DD HH24:MI:SS'), 'p')
```

```
INSERT INTO "ATTENDANCE" (ATTENDANCE_ID, STUDENT_ID, CURRENT_DATE, STATUS) VALUES ('2', '1', TO_DATE('2022-12-06 12:20:42', 'YYYY-MM-DD HH24:MI:SS'), 'np')
```

```
create table book(  
book_id int primary key,
```

```
name varchar(20),  
description varchar(20))  
  
INSERT INTO book(  
book_id,  
name,  
description)  
VALUES(1, 'Intro to python', 'A book for python')
```

```
INSERT INTO book(  
book_id,  
name,  
description)  
VALUES(2, 'Intro to Data', 'Data concepts')
```

```
create table library(  
library_id int primary key,  
student_id int,  
book_id int,  
status varchar(20),  
CONSTRAINT fk_student_check1  
    FOREIGN KEY (student_id)  
    REFERENCES student(Student_id),  
CONSTRAINT fk_book_check1  
    FOREIGN KEY (book_id)  
    REFERENCES Book(book_id)  
)
```

```
INSERT INTO library(
library_id,
student_id,
book_id,
status
)
VALUES (11, 1, 2, 'Given')
VALUES (12, 2, 1, 'Returned')
```

```
create table exam_type(
exam_type_id int primary key,
type varchar(20),
description varchar(100))
```

```
INSERT INTO "EXAM_TYPE" (EXAM_TYPE_ID, TYPE, DESCRIPTION) VALUES ('1', 'mid term', 'first exam')
INSERT INTO "EXAM_TYPE" (EXAM_TYPE_ID, TYPE, DESCRIPTION) VALUES ('2', 'last tern', 'final exam')
```

```
create table staff(
staff_id int primary key,
first_name varchar(20) not null,
last_name varchar(20),
phone varchar(10),
email varchar(30),
```

```
    dob date not null,  
    designation varchar(20)  
)
```

```
INSERT INTO "STAFF" (STAFF_ID, FIRST_NAME, LAST_NAME, PHONE, EMAIL, DOB, DESIGNATION)  
VALUES ('1', 'aman', 'kaul', '6478324567', 'aman@gmail', TO_DATE('1996-10-04 12:29:17', 'YYYY-MM-  
DD HH24:MI:SS'), 'teacher')
```

```
INSERT INTO "STAFF" (STAFF_ID, FIRST_NAME, LAST_NAME, PHONE, EMAIL, DOB, DESIGNATION)  
VALUES ('2', 'athul', 'madav', '6478354672', 'athul123@mail.com', TO_DATE('1995-12-07 12:30:28',  
'YYYY-MM-DD HH24:MI:SS'), 'teacher')
```

```
INSERT INTO "STAFF" (STAFF_ID, FIRST_NAME, LAST_NAME, PHONE, EMAIL, DOB, DESIGNATION)  
VALUES ('3', 'hari', 'mohan', '5678324567', 'harimohan@gmail.com', TO_DATE('1998-08-11 12:31:35',  
'YYYY-MM-DD HH24:MI:SS'), 'head of department')
```

```
create table grade(  
grade_id int primary key,  
name varchar(20),  
description varchar(100))
```

```
INSERT INTO "GRADE" (GRADE_ID, NAME, DESCRIPTION) VALUES ('1', 'th grade', 'high school')  
INSERT INTO "GRADE" (GRADE_ID, NAME, DESCRIPTION) VALUES ('2', '9th grade', 'high school')
```

```
create table student_class(  
class_id int primary key,  
student_id int,  
constraint check_student_class  
foreign key(student_id)
```

```
references student(Student_id)
)

INSERT INTO "STUDENT_CLASS" (CLASS_ID, STUDENT_ID) VALUES ('1', '1')
INSERT INTO "STUDENT_CLASS" (CLASS_ID, STUDENT_ID) VALUES ('2', '2')
```

```
Create table class_enrolled(
    class_enrolled_id int primary key,
    grade_id int,
    staff_id int,
    year_of_attempt date,
    constraint check_grade_class
        foreign key(grade_id)
        references grade(grade_id),
    constraint check_staff_class
        foreign key(staff_id)
        references staff(staff_id)
)

INSERT INTO "CLASS_ENROLLED" (CLASS_ENROLLED_ID, GRADE_ID, STAFF_ID, YEAR_OF_ATTEMPT)
VALUES ('1', '1', '1', TO_DATE('2022-12-06 12:35:33', 'YYYY-MM-DD HH24:MI:SS'))
INSERT INTO "CLASS_ENROLLED" (CLASS_ENROLLED_ID, GRADE_ID, STAFF_ID, YEAR_OF_ATTEMPT)
VALUES ('2', '2', '2', TO_DATE('2022-12-06 12:35:38', 'YYYY-MM-DD HH24:MI:SS'))
```

```
create table course(
```

```
course_id int primary key,  
grade_id int,  
name varchar(20),  
description varchar(100),  
credits int,  
constraint check_grade  
foreign key(grade_id)  
references grade(grade_id)  
)
```

```
INSERT INTO "COURSE" (COURSE_ID, GRADE_ID, NAME, DESCRIPTION, CREDITS) VALUES ('1', '1', 'data base', 'basic sql queries', '10')
```

```
INSERT INTO "COURSE" (COURSE_ID, GRADE_ID, NAME, DESCRIPTION, CREDITS) VALUES ('2', '2', 'python', 'basic python gaming', '10')
```

```
create table exam_result(  
exam_result_id int primary key,  
student_id int,  
course_id int,  
marks int,  
constraint check_course  
foreign key(course_id)  
references course(course_id),  
constraint check_student_exam_result  
foreign key(student_id)  
references student(student_id)  
)
```

```
INSERT INTO "EXAM_RESULT" (EXAM_RESULT_ID, STUDENT_ID, COURSE_ID, MARKS) VALUES ('1', '1', '1', '10')
```

```
INSERT INTO "EXAM_RESULT" (EXAM_RESULT_ID, STUDENT_ID, COURSE_ID, MARKS) VALUES ('2', '1', '2', '9')
```

```
create table exam (
exam_val_id int primary key,
course_id int,
exam_type_id int,
student_id int ,
start_date date not null,
constraint check_student_exam
foreign key(student_id)
references student(student_id),
constraint check_course_exam
foreign key(course_id)
references course(course_id),
constraint check_exam_type_id
foreign key(exam_type_id)
references exam_type(exam_type_id)
)
```

```
INSERT INTO "EXAM" (EXAM_VAL_ID, COURSE_ID, EXAM_TYPE_ID, STUDENT_ID, START_DATE)
VALUES ('1', '1', '1', '1', TO_DATE('2022-12-08 12:50:24', 'YYYY-MM-DD HH24:MI:SS'))
```

```
INSERT INTO "EXAM" (EXAM_VAL_ID, COURSE_ID, EXAM_TYPE_ID, STUDENT_ID, START_DATE)
VALUES ('2', '2', '1', '1', TO_DATE('2022-12-09 12:50:53', 'YYYY-MM-DD HH24:MI:SS'))
```

```
INSERT INTO "EXAM" (EXAM_VAL_ID, COURSE_ID, EXAM_TYPE_ID, STUDENT_ID, START_DATE)
VALUES ('3', '1', '1', '2', TO_DATE('2022-12-08 12:51:38', 'YYYY-MM-DD HH24:MI:SS'))
```

```
INSERT INTO "EXAM" (EXAM_VAL_ID, COURSE_ID, EXAM_TYPE_ID, STUDENT_ID, START_DATE)
VALUES ('4', '2', '1', '2', TO_DATE('2022-12-09 12:52:08', 'YYYY-MM-DD HH24:MI:SS'))
```

## NORMALIZED TABLES

### Student & Address

STUDENT_ID	ADDRESS_ID	FIRST_NAME	LAST_NAME	GUARDIAN_NAME	DOB	MOBILE	EMAIL	JOINING_DATE
1	1	akhilesh	saju	saju	05-DEC-97	6478325376	akhileshsaju@gmail.com	06-DEC-22
2	2	praveen	saji	saji	03-JUN-97	6475687689	praveen@hotmail.com	06-DEC-22

[Download CSV](#)

2 rows selected.

ADDRESS_ID	APT_NO	STREET	CITY	ZIPCODE
2	69	Williams Parkway	Brmapton	L3X2C1
1	14	McLaughlin	Brmapton	L5X6W1

#### ▪ 1NF:

- ✓ Primary Key is defined.
- ✓ No multivalued Columns and repeating groups found.
- ✓ Hence, they are in 1NF.

#### ▪ 2NF:

- ✓ No partial dependencies found in all the tables.
- ✓ Hence, they are in 2NF.

#### ▪ 3NF:

- ✓ No transitive dependencies found in all the tables.
- ✓ Hence, they are in 3NF.

### Student\_Council

STUDENT_COUNCIL_ID	STUDENT_ID	DESIGNATION
1	1	m
2	2	p

#### ▪ 1NF:

- ✓ Primary Key is defined.
- ✓ No multivalued Columns and repeating groups found.

- ✓ Hence, they are in 1NF.

▪ **2NF:**

- ✓ No partial dependencies found in all the tables.
- ✓ Hence, they are in 2NF.

▪ **3NF:**

- ✓ No transitive dependencies found in all the tables.
- ✓ Hence, they are in 3NF.

## Attendance

---

ATTENDANCE_ID	STUDENT_ID	CURRENT_DATE	STATUS
1	2	06-DEC-22	p
2	1	06-DEC-22	np

▪ **1NF:**

- ✓ Primary Key is defined.
- ✓ No multivalued Columns and repeating groups found.
- ✓ Hence, they are in 1NF.

▪ **2NF:**

- ✓ No partial dependencies found in all the tables.
- ✓ Hence, they are in 2NF.

▪ **3NF:**

- ✓ No transitive dependencies found in all the tables.
- ✓ Hence, they are in 3NF.

## Book

BOOK_ID	NAME	DESCRIPTION
1	Intro to python	A book for python
2	Intro to Data	Data concepts

■ **1NF:**

- ✓ Primary Key is defined.
- ✓ No multivalued Columns and repeating groups found.
- ✓ Hence, they are in 1NF.

■ **2NF:**

- ✓ No partial dependencies found in all the tables.
- ✓ Hence, they are in 2NF.

■ **3NF:**

- ✓ No transitive dependencies found in all the tables.
- ✓ Hence, they are in 3NF.

## Library

LIBRARY_ID	STUDENT_ID	BOOK_ID	STATUS
11	1	2	Given
12	2	1	Returned

■ **1NF:**

- ✓ Primary Key is defined.
- ✓ No multivalued Columns and repeating groups found.
- ✓ Hence, they are in 1NF.

■ **2NF:**

- ✓ No partial dependencies found in all the tables.
- ✓ Hence, they are in 2NF.

▪ **3NF:**

- ✓ No transitive dependencies found in all the tables.
- ✓ Hence, they are in 3NF.

### Exam\_Type

EXAM_TYPE_ID	TYPE	DESCRIPTION
1	mid term	first exam
2	last tern	final exam

▪ **1NF:**

- ✓ Primary Key is defined.
- ✓ No multivalued Columns and repeating groups found.
- ✓ Hence, they are in 1NF.

▪ **2NF:**

- ✓ No partial dependencies found in all the tables.
- ✓ Hence, they are in 2NF.

▪ **3NF:**

- ✓ No transitive dependencies found in all the tables.
- ✓ Hence, they are in 3NF.

## Staff

STAFF_ID	FIRST_NAME	LAST_NAME	PHONE	EMAIL	DOB	DESIGNATION
2	athul	madav	6478354672	athul123@mail.com	07-DEC-95	teacher
1	aman	kaul	6478324567	aman@gmail	04-OCT-96	teacher
3	hari	mohan	5678324567	harimohan@gmail.com	11-AUG-98	head of department

▪ **1NF:**

- ✓ Primary Key is defined.
- ✓ No multivalued Columns and repeating groups found.
- ✓ Hence, they are in 1NF.

▪ **2NF:**

- ✓ No partial dependencies found in all the tables.
- ✓ Hence, they are in 2NF.

▪ **3NF:**

- ✓ No transitive dependencies found in all the tables.
- ✓ Hence, they are in 3NF.

## Grade

GRADE_ID	NAME	DESCRIPTION
1	th grade	high school
2	9th grade	high school

▪ **1NF:**

- ✓ Primary Key is defined.
- ✓ No multivalued Columns and repeating groups found.
- ✓ Hence, they are in 1NF.

▪ **2NF:**

- ✓ No partial dependencies found in all the tables.
- ✓ Hence, they are in 2NF.

▪ **3NF:**

- ✓ No transitive dependencies found in all the tables.
- ✓ Hence, they are in 3NF.

## Class

---

CLASS_ID	STUDENT_ID
1	1
2	2

▪ **1NF:**

- ✓ Primary Key is defined.
- ✓ No multivalued Columns and repeating groups found.
- ✓ Hence, they are in 1NF.

▪ **2NF:**

- ✓ No partial dependencies found in all the tables.
- ✓ Hence, they are in 2NF.

▪ **3NF:**

- ✓ No transitive dependencies found in all the tables.
- ✓ Hence, they are in 3NF.

## Class\_Enrolled

CLASS_ENROLLED_ID	GRADE_ID	STAFF_ID	YEAR_OF_ATTEMPT
1	1	1	06-DEC-22
2	2	2	06-DEC-22

▪ **1NF:**

- ✓ Primary Key is defined.
- ✓ No multivalued Columns and repeating groups found.
- ✓ Hence, they are in 1NF.

▪ **2NF:**

- ✓ No partial dependencies found in all the tables.
- ✓ Hence, they are in 2NF.

▪ **3NF:**

- ✓ No transitive dependencies found in all the tables.
- ✓ Hence, they are in 3NF.

## Course

COURSE_ID	GRADE_ID	NAME	DESCRIPTION	CREDITS
1	1	data base	basic sql queries	10
2	2	python	basic python gaming	10

▪ **1NF:**

- ✓ Primary Key is defined.
- ✓ No multivalued Columns and repeating groups found.
- ✓ Hence, they are in 1NF.

▪ **2NF:**

- ✓ No partial dependencies found in all the tables.
- ✓ Hence, they are in 2NF.

▪ **3NF:**

- ✓ No transitive dependencies found in all the tables.
- ✓ Hence, they are in 3NF.

## Exam Result

---

EXAM_RESULT_ID	STUDENT_ID	COURSE_ID	MARKS
1	1	1	10
2	1	2	9

▪ **1NF:**

- ✓ Primary Key is defined.
- ✓ No multivalued Columns and repeating groups found.
- ✓ Hence, they are in 1NF.

▪ **2NF:**

- ✓ No partial dependencies found in all the tables.
- ✓ Hence, they are in 2NF.

▪ **3NF:**

- ✓ No transitive dependencies found in all the tables.
- ✓ Hence, they are in 3NF.

## Exam\_Val

EXAM_VAL_ID	COURSE_ID	EXAM_TYPE_ID	STUDENT_ID	START_DATE
3	1	1	2	08-DEC-22
1	1	1	1	08-DEC-22
2	2	1	1	09-DEC-22
4	2	1	2	09-DEC-22

■ 1NF:

- ✓ Primary Key is defined.
- ✓ No multivalued Columns and repeating groups found.
- ✓ Hence, they are in 1NF.

■ 2NF:

- ✓ No partial dependencies found in all the tables.
- ✓ Hence, they are in 2NF.

■ 3NF:

- ✓ No transitive dependencies found in all the tables.
- ✓ Hence, they are in 3NF.

## PHYSICAL DATABASE DESIGN:

### Creating the table:

Student Table created.

The screenshot shows the Oracle SQL Developer interface. In the central workspace, there is a 'Worksheet' tab with the following SQL code:

```
create table Student(
    Student_id int primary key,
    address_id int not null,
    first_name varchar(20) not null,
    last_name varchar(20) not null,
    Guardian_name varchar(20),
    dob date not null,
    mobile varchar(15),
    email varchar(50),
    Joining_date date not null,
    CONSTRAINT fk_address_id
        FOREIGN KEY (address_id)
            REFERENCES ADDRESS(address_id)
);

drop table Student
```

Below the worksheet, in the 'Script Output' tab, the results of the execution are shown:

```
Table ADDRESS created.  
Table STUDENT created.
```

A red box highlights the right side of the 'Script Output' tab, containing the following bullet points:

- Script for creating Student Table with CONSTRAINTS.
- SQL table "Student" created successfully.

Structure of the student table.

Oracle SQL Developer - Assignment.sd

File Edit View Navigate Run Team Tools Window Help

STUDENT - Structure

No Structure

Connections

- ROLLINGPARAMETERS
- ROLLINGPLAN
- ROLLINGSTATISTICS
- ROLLINGSTATUS
- SCHEDULER\_JOB\_ARGS\_TBL
- SQLOPLUS\_PRODUCT\_PROFILE
- STAFF
- STUDENT**
- STUDENT\_ID
- ADDRESS\_ID

Reports

- All Reports
- Analytic View Reports
- Data Dictionary Reports
- Data Modeler Reports
- OLAP Reports
- TimesTen Reports
- User Defined Reports

Actions...

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 STUDENT_ID	NUMBER(38,0)	No	(null)	1 (null)	
2 ADDRESS_ID	NUMBER(38,0)	No	(null)	2 (null)	
3 FIRST_NAME	VARCHAR2(20 BYTE)	No	(null)	3 (null)	
4 LAST_NAME	VARCHAR2(20 BYTE)	No	(null)	4 (null)	
5 GUARDIAN_NAME	VARCHAR2(20 BYTE)	Yes	(null)	5 (null)	
6 DOB	DATE	No	(null)	6 (null)	
7 MOBILE	VARCHAR2(10 BYTE)	Yes	(null)	7 (null)	
8 EMAIL	VARCHAR2(50 BYTE)	Yes	(null)	8 (null)	
9 JOINING_DATE	DATE	No	(null)	9 (null)	

- Structure of the Student table with DATA TYPES & Comments.

## Address Table created:

Oracle SQL Developer : project

File Edit View Navigate Run Source Team Tools Window Help

Connections

- Als
- Assignment
- project
- Tables (Filtered)
- Views
- Indexes
- Packages
- Procedures
- Functions
- Operators
- Queues
- Queues Tables
- Triggers
- Types
- Sequences
- Materialized Views
- Materialized View Logs

Reports

- All Reports
- Analytic View Reports
- Data Dictionary Reports
- Data Modeler Reports
- OLAP Reports
- TimesTen Reports
- User Defined Reports

Worksheet - Query Builder

```
create table ADDRESS(
    address_id int primary key,
    apt_no varchar(20),
    street varchar(20),
    city varchar(20),
    zipcode varchar(20)
)

create table Student(
    Student_id int primary key,
    address_id int not null,
    First_name varchar(20) not null,
    Last_name varchar(20) not null,
    Guardian_name varchar(20),
    dob date not null,
    Mobile varchar(10),
    email varchar(50)
)
```

Script Output X

Task completed in 0.246 seconds

Table ADDRESS created.

- Script for creating ADDRESS Table with primary key CONSTRAINT.
- SQL table “ADDRESS” created successfully.

The screenshot shows the Oracle SQL Developer interface. The main window displays the structure of the 'ADDRESS' table. The table has five columns: ADDRESS\_ID (NUMBER(38,0)), APT\_NO (VARCHAR2(20 BYTE)), STREET (VARCHAR2(20 BYTE)), CITY (VARCHAR2(20 BYTE)), and ZIPCODE (VARCHAR2(20 BYTE)). The 'APT\_NO' column is nullable, while the others are not. The 'Comments' column shows '(null)' for all rows. A red arrow points from the table structure towards a callout box.

COLUMN_NAME	DATA_TYPE	NULLLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ADDRESS_ID	NUMBER(38,0)	No	(null)	1	(null)
2 APT_NO	VARCHAR2(20 BYTE)	Yes	(null)	2	(null)
3 STREET	VARCHAR2(20 BYTE)	Yes	(null)	3	(null)
4 CITY	VARCHAR2(20 BYTE)	Yes	(null)	4	(null)
5 ZIPCODE	VARCHAR2(20 BYTE)	Yes	(null)	5	(null)

- Structure of the ADDRESS table with DATA TYPES & Column IDs.

## Attendance table created:

The screenshot shows the Oracle SQL Developer interface with a query editor containing the script for creating the 'Attendance' table. The script includes constraints like a primary key on Attendance\_id, a foreign key constraint fk\_student\_check referencing the student table, and a check constraint on Status. Below the script, the 'Script Output' window shows the successful creation of four tables: ADDRESS, STUDENT, STUDENT\_COUNCIL, and ATTENDANCE. A red arrow points from the 'Script Output' window towards a callout box.

```

create table Attendance(
    Attendance_id int primary key,
    Student_Id int,
    Current_date date not null,
    Status varchar(20) CONSTRAINT check_status_of_attendance
        CHECK (Status in ('present','absent')) disable,
    CONSTRAINT fk_student_check
        FOREIGN KEY (Student_Id)
        REFERENCES student(Student_id)
);

create table book(
    book_id int primary key,
    name varchar(20),
);

```

- Script for creating Attendance Table with CONSTRAINTS.
- SQL table "Attendance" created successfully.

## Attendance table structure.

The screenshot shows the Oracle SQL Developer interface. In the center, the 'ATTENDANCE - Structure' window is open, displaying the table structure. The columns are:

COLUMN_NAME	DATA_TYPE	NNULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
ATTENDANCE_ID	NUMBER(38,0)	No	(null)	1	(null)
STUDENT_ID	NUMBER(38,0)	Yes	(null)	2	(null)
CURRENT_DATE	DATE	No	(null)	3	(null)
STATUS	VARCHAR2(20 BYTE)	Yes	(null)	4	(null)

To the left, the 'Connections' sidebar shows a tree structure with various database objects, including 'ATTENDANCE'. A red arrow points from the 'ATTENDANCE' node in the tree to the table structure window. To the right, a callout box contains the following text:

- Structure of the Attendance table with DATA TYPES & Column IDs.

## Book table created:

The screenshot shows the Oracle SQL Developer interface. In the center, the 'Worksheet' window displays the SQL script for creating the 'book' table:

```
create table book(
    book_id int primary key,
    name varchar(20),
    description varchar(20))
```

Below the worksheet, the 'Script Output' window shows the results of the execution:

```
Table ADDRESS created.  
Table STUDENT created.  
Table STUDENT_COUNCIL created.  
Table ATTENDANCE created.  
Table BOOK created.
```

A red arrow points from the 'book' table definition in the worksheet to the 'Script Output' window. Another red arrow points from the 'book' table definition to a callout box on the right. A third red arrow points from the 'Script Output' window to the same callout box. The callout box contains the following text:

- Script for creating Book Table with CONSTRAINTS.
- SQL table "Book" created successfully.

## Structure for book table:

The screenshot shows the Oracle SQL Developer interface. The main window displays the 'BOOK - Structure' tab, which shows the columns of the 'BOOK' table. The columns are:

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 BOOK_ID	NUMBER(18,0)	No	(null)	1	(null)
2 NAME	VARCHAR2(20 BYTE)	Yes	(null)	2	(null)
3 DESCRIPTION	VARCHAR2(20 BYTE)	Yes	(null)	3	(null)

A red arrow points from the 'Structure of the Book table with DATA TYPES & Column IDs.' text box to the 'BOOK - Structure' tab.

- Structure of the Book table with DATA TYPES & Column IDs.

## Table created for class enrolled:

The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab contains the SQL script for creating the 'class\_enrolled' table:

```
references student(Student_id)
)
Create table class_enrolled(
class_enrolled_id int primary key,
grade_id int,
staff_id int,
year_of_attempt date,
constraint check_grade_class
foreign key(grade_id)
references grade(grade_id),
constraint check_staff_class
foreign key(staff_id)
references staff(staff_id)
)
create table course(

```

The 'Script Output' tab shows the results of the execution:

```
Table EXAM_TYPE created.
Table STAFF created.
Table GRADE created.
Table STUDENT_CLASS created.
Table CLASS_ENROLLED created.
```

A red arrow points from the 'Script for creating Class Table with CONSTRAINTS.' text box to the 'Worksheet' tab. Another red arrow points from the 'SQL table "Class Enrolled" created.' text box to the 'Script Output' tab.

- Script for creating Class Table with CONSTRAINTS.
- SQL table "Class Enrolled" created.

## Structure of students enrolled:

The screenshot shows the Oracle SQL Developer interface. The main window displays the structure of the 'CLASS\_ENROLLED' table. The table has four columns: 'CLASS\_ENROLLED\_ID' (NUMBER(38,0) No), 'GRADE\_ID' (NUMBER(38,0) Yes), 'STAFF\_ID' (NUMBER(38,0) Yes), and 'YEAR\_OF\_ATTEMPT' (DATE Yes). A red arrow points from the table structure area to a callout box.

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 CLASS_ENROLLED_ID	NUMBER (38,0)	No	(null)	1	(null)
2 GRADE_ID	NUMBER (38,0)	Yes	(null)	2	(null)
3 STAFF_ID	NUMBER (38,0)	Yes	(null)	3	(null)
4 YEAR_OF_ATTEMPT	DATE	Yes	(null)	4	(null)

**Structure of the Students Enrolled table with DATA TYPES & Column IDs.**

## Student council table created:

The screenshot shows the Oracle SQL Developer interface with the 'Worksheet' tab active. The code in the worksheet creates the 'Student\_Council' table with constraints and then creates the 'Attendance' table. Red arrows point from the code area to a callout box.

```
drop table Student;

create table Student_Council(
student_council_id int primary key,
student_id int,
Designation varchar(20),
CONSTRAINT fk_student
    FOREIGN KEY (Student_id)
    REFERENCES student(Student_id)
);

select * from Attendance

create table Attendance(
```

**Script for creating Student Council Table with CONSTRAINTS.**

**SQL table "Student Council" created.**

Oracle SQL Developer - Structure

Assignment.sql | Welcome Page | project | STUDENT\_COUNCIL

No Structure

Connections

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 STUDENT_COUNCIL_ID	NUMBER(38,0)	No	(null)	1	(null)
2 STUDENT_ID	NUMBER(38,0)	Yes	(null)	2	(null)
3 DESIGNATION	VARCHAR2(20 BYTE)	Yes	(null)	3	(null)

Reports

- All Reports
- Analytic View Reports
- Data Dictionary Reports
- Data Modeler Reports
- OLAP Reports
- TimexTen Reports
- User Defined Reports

- Structure of the Student Council table with DATA TYPES & Column IDs.

## Table created for Exam Type:

Oracle SQL Developer : project

File Edit View Navigate Run Source Team Window Help

Connections

Worksheet

```

    REFERENCES student(Student_id),
    CONSTRAINT fk_book_check1
    FOREIGN KEY (book_id)
    REFERENCES Book(book_id)

)

create table exam_type(
exam_type_id int primary key,
type varchar(20),
description varchar(100)
)

create table staff(
staf_id int primary key,
first_name varchar(20) not null,
last_name varchar(20),

```

Script Output

Task completed in 0.046 seconds

Table STUDENT\_COUNCIL created.

Table ATTENDANCE created.

Table BOOK created.

Table LIBRARY created.

Table EXAM\_TYPE created.

- Script for creating Exam Type Table with CONSTRAINTS.
- SQL table "Exam Type" created.

## Structure for exam type:

The screenshot shows the Oracle SQL Developer interface. In the center, the 'EXAM\_TYPE - Structure' window is open, displaying the columns of the EXAM\_TYPE table. The columns are:

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 EXAM_TYPE_ID	NUMBER(18,0)	No	(null)	1	(null)
2 TYPE	VARCHAR2(20 BYTE)	Yes	(null)	2	(null)
3 DESCRIPTION	VARCHAR2(100 BYTE)	Yes	(null)	3	(null)

To the left, the 'Connections' sidebar shows the 'EXAM\_RESULT' and 'EXAM\_TYPE' tables. A red arrow points from the table structure window to a callout box containing the following text:

- Structure of the Exam Type table with DATA TYPES & Column IDs.

## Table created for Exam:

The screenshot shows the Oracle SQL Developer interface. In the center, the 'Worksheet' window displays the SQL script used to create the EXAM table:

```
references student(student_id)
)
select * from course
create table exam (
exam_val_id int primary key,
course_id int,
exam_type_id int,
student_id int,
start_date date not null,
constraint check_student_exam
foreign key(student_id)
references student(student_id),
constraint check_course_exam
foreign key(course_id)
references course(course_id),
constraint check_exam_type_id
foreign key(exam_type_id)
```

The 'Script Output' window shows the results of the execution:

```
Table COURSE created.

Table EXAM_RESULT created.

Table EXAM created.
```

To the left, the 'Connections' sidebar shows the 'project' connection with its tables. A red arrow points from the worksheet window to a callout box containing the following text:

- Script for creating Exam Table with CONSTRAINTS.
- SQL table "Exam" created successfully.

## Table for library:

The screenshot shows the Oracle SQL Developer interface. In the central workspace, a script is being run to create a table named 'library'. The code includes primary key and foreign key constraints. The 'Script Output' pane at the bottom shows the successful creation of various tables: STUDENT, STUDENT\_COUNCIL, ATTENDANCE, BOOK, and LIBRARY.

```
create table library(
    library_id int primary key,
    student_id int,
    book_id int,
    status varchar(20),
    CONSTRAINT fk_student_check1
        FOREIGN KEY (student_id)
            REFERENCES student(Student_id),
    CONSTRAINT fk_book_check1
        FOREIGN KEY (book_id)
            REFERENCES Book(book_id)
```

```
create table exam type()
```

Table STUDENT created.  
Table STUDENT\_COUNCIL created.  
Table ATTENDANCE created.  
Table BOOK created.  
Table LIBRARY created.

- Script for creating Library Table with CONSTRAINTS.
- SQL table "Library" created successfully.

## Structure for Library:

The screenshot shows the Oracle SQL Developer interface with the 'LIBRARY - Structure' tab selected. It displays the columns of the LIBRARY table, including their names, data types, nullability, default values, and column IDs.

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 LIBRARY_ID	NUMBER(38,0)	No	(null)	1 (null)	
2 STUDENT_ID	NUMBER(38,0)	Yes	(null)	2 (null)	
3 BOOK_ID	NUMBER(38,0)	Yes	(null)	3 (null)	
4 STATUS	VARCHAR2(20 BYTE)	Yes	(null)	4 (null)	

- Structure of the Library table with DATA TYPES & Column IDs.

## Table created for exam result:

The screenshot shows the Oracle SQL Developer interface. In the central workspace, a SQL script is being run to create the EXAM\_RESULT table. The script includes constraints such as primary key, foreign keys, and a check constraint. The output window shows the successful creation of the COURSE and EXAM\_RESULT tables. A red arrow points from the text box below to the SQL code in the workspace.

```
create table exam_result(
exam_result_id int primary key,
student_id int,
course_id int,
marks int,
constraint check_course
foreign key(course_id)
references course(course_id),
constraint check_student_exam_result
foreign key(student_id)
references student(student_id)

select * from course
create table exam {
```

Table COURSE created.  
Table EXAM\_RESULT created.

- Script for creating Exam Result Table with CONSTRAINTS.
- SQL table "Exam Result" created.

## Structure for exam result:

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the database schema, specifically the EXAM\_RESULT table structure. The right pane shows the detailed structure of the EXAM\_RESULT table, including four columns: EXAM\_RESULT\_ID, STUDENT\_ID, COURSE\_ID, and MARKS, each with their respective data types (NUMBER) and constraints (NOT NULL). A red arrow points from the text box below to the column details in the right pane.

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 EXAM_RESULT_ID	NUMBER(35,0)	No	(null)	1	(null)
2 STUDENT_ID	NUMBER(35,0)	Yes	(null)	2	(null)
3 COURSE_ID	NUMBER(35,0)	Yes	(null)	3	(null)
4 MARKS	NUMBER(35,0)	Yes	(null)	4	(null)

- Structure of the Exam Result table with DATA TYPES & Column IDs.

## Table created for grades:

The screenshot shows the Oracle SQL Developer interface. In the central workspace, a script is being run in the Worksheet tab, creating several tables including a 'GRADE' table with constraints. The 'Script Output' pane shows the results of the execution, indicating that various tables like 'BOOK', 'LIBRARY', 'EXAM\_TYPE', 'STAFF', and 'GRADE' have been successfully created. A red arrow points from the 'Script Output' pane to a callout box on the right.

- Script for creating Grades Table with CONSTRAINTS.
- SQL table "Grades" created.

## Structure for grade table:

The screenshot shows the Oracle SQL Developer interface. The central workspace displays the structure of the 'GRADE' table, specifically its columns and their data types. A red arrow points from the 'GRADE' table structure to a callout box on the right.

- Structure of the Grade table with DATA TYPES & Column IDs.

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 GRADE_ID	NUMBER(38,0)	No	(null)	1 (null)	
2 NAME	VARCHAR2(20 BYTE)	Yes	(null)	2 (null)	
3 DESCRIPTION	VARCHAR2(100 BYTE)	Yes	(null)	3 (null)	

## Table created for student class:

The screenshot shows the Oracle SQL Developer interface. In the central Worksheet pane, a SQL script is being run to create a table named 'STUDENT\_CLASS'. The script includes constraints such as a primary key on 'CLASS\_ID' and a foreign key on 'STUDENT\_ID' referencing the 'STUDENT' table. The 'Script Output' pane shows the execution results, indicating that several tables were created: LIBRARY, EXAM\_TYPE, STAFF, GRADE, and STUDENT\_CLASS. A red arrow points from the 'Script Output' text to a callout box.

```
name varchar(20),  
description varchar(100))  
  
create table student_class(  
class_id int primary key,  
student_id int,  
constraint check_student_class  
foreign key(student_id)  
references student(Student_id)  
  
Create table class_enrolled(  
class_enrolled_id int primary key,  
grade_id int,  
staff_id int,  
year of attempt date,
```

Table LIBRARY created.  
Table EXAM\_TYPE created.  
Table STAFF created.  
Table GRADE created.  
Table STUDENT\_CLASS created.

- Script for creating Student Class Table with CONSTRAINTS.
- SQL table “Student Class” created.

## Structure for Student class:

The screenshot shows the Oracle SQL Developer interface with the 'STUDENT\_CLASS - Structure' tab selected. It displays the columns of the 'STUDENT\_CLASS' table, including 'CLASS\_ID' (NUMBER(38,0) No) and 'STUDENT\_ID' (NUMBER(38,0) Yes). A red arrow points from the table structure to a callout box.

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 CLASS_ID	NUMBER(38,0)	No	(null)	1	(null)
2 STUDENT_ID	NUMBER(38,0)	Yes	(null)	2	(null)

- Structure of the Student Class table with DATA TYPES & Column IDs.

### 3. Physical Database Design

- Create **a table of your choice** and do the following operations
- CREATE (enter) 5 records to the table using SQL Script
- UPDATE **one or more** records (Using Script)
- DELETE **one or more** records (Using Script)
- DROP the table you created.
- Do not use your project tables for this activity.

➤ [Create Table](#)

The screenshot shows the Oracle Live SQL interface. The left sidebar has a dark theme with white text. The "SQL Worksheet" section is selected. The main area is titled "SQL Worksheet" and contains the following SQL code:

```
1 CREATE TABLE Employee
2 (
3     EmployeeNo char(4),
4     EmployeeName varchar2(30),
5     EmployeeSal number(10,2),
6     EmployeeCity varchar2(30),
7     EmployeeDob date
8 );
9
```

Below the code, a message says "Table created." There are several buttons at the top right: "Clear", "Find", "Actions", "Save", and "Run". The top right corner shows the user's email: "fahadismail053@gmail.com".

➤ [Enter 5 records to the table](#)

Live SQL

SQL Worksheet

```

1 INSERT INTO Employee(EmployeeNo,EmployeeName,EmployeeSal,EmployeeCity,EmployeeDob)
2 values('1', 'Rudhar', 5000, 'Mississauga','23-DEC-1998');
3
4 INSERT INTO Employee(EmployeeNo,EmployeeName,EmployeeSal,EmployeeCity,EmployeeDob)
5 values('2', 'Reea', 6000, 'Brampton','15-JAN-1999');
6
7 INSERT INTO Employee
8 values('3', 'Fahad', 4000, 'Ajax','20-MAR-1998');
9
10 INSERT INTO Employee
11 values('4', 'Praveen', 8000, 'Toronto','05-NOV-1997');
12
13 INSERT INTO Employee
14 values('5', 'Akhilesh', 9000, 'Oshawa','13-FEB-1996');

```

1 row(s) inserted.

Live SQL

SQL Worksheet

```

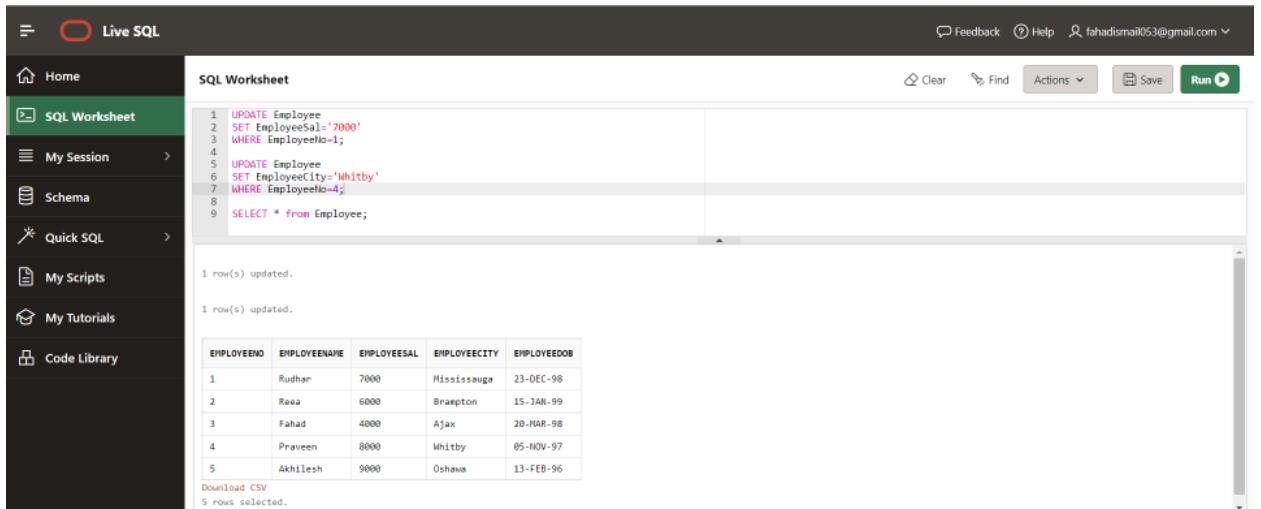
1 INSERT INTO Employee(EmployeeNo,EmployeeName,EmployeeSal,EmployeeCity,EmployeeDob)
2 values('1', 'Rudhar', 5000, 'Mississauga','23-DEC-1998');
3
4 INSERT INTO Employee(EmployeeNo,EmployeeName,EmployeeSal,EmployeeCity,EmployeeDob)
5 values('2', 'Reea', 6000, 'Brampton','15-JAN-1999');
6
7 INSERT INTO Employee
8 values('3', 'Fahad', 4000, 'Ajax','20-MAR-1998');
9
10 INSERT INTO Employee
11 values('4', 'Praveen', 8000, 'Toronto','05-NOV-1997');
12
13 INSERT INTO Employee
14 values('5', 'Akhilesh', 9000, 'Oshawa','13-FEB-1996');
15
16 select * from Employee

```

EMPLOYEEENO	EMPLOYEEENAME	EMPLOYEESAL	EMPLOYEECITY	EMPLOYEEDOB
1	Rudhar	5000	Mississauga	23-DEC-98
2	Reea	6000	Brampton	15-JAN-99
3	Fahad	4000	Ajax	20-MAR-98
4	Praveen	8000	Toronto	05-NOV-97
5	Akhilesh	9000	Oshawa	13-FEB-96

Download CSV  
5 rows selected.

➤ Update One or More records:



The screenshot shows the Live SQL interface with the following details:

- SQL Worksheet:**

```

1 UPDATE Employee
2 SET EmployeeSal='7000'
3 WHERE EmployeeNo=1;
4
5 UPDATE Employee
6 SET EmployeeCity='Whitby'
7 WHERE EmployeeNo=4;
8
9 SELECT * from Employee;

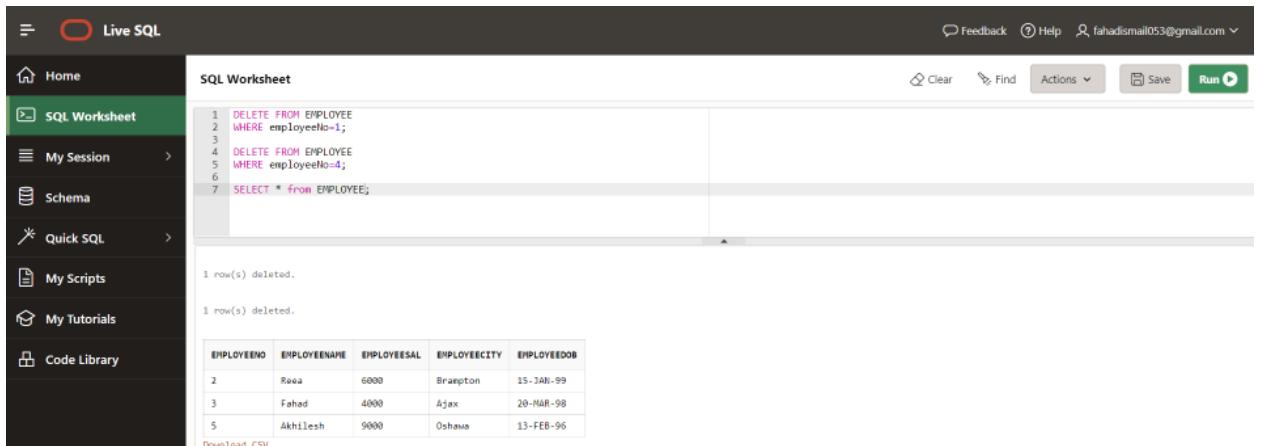
```
- Output:**

1 row(s) updated.

1 row(s) updated.
- Table:**

EMPLOYEEID	EMPLOYEENAME	EMPLOYEESSAL	EMPLOYEECITY	EMPLOYEEDOB
1	Kudhar	7000	Mississauga	23-DEC-98
2	Rhea	6000	Brampton	15-JAN-99
3	Fahad	4000	Ajax	20-MAR-98
4	Praveen	8000	Whitby	05-NOV-97
5	Akhilesh	9000	Oshawa	13-FEB-96

➤ Delete One or More Records:



The screenshot shows the Live SQL interface with the following details:

- SQL Worksheet:**

```

1 DELETE FROM EMPLOYEE
2 WHERE employeeid=1;
3
4 DELETE FROM EMPLOYEE
5 WHERE employeeid=4;
6
7 SELECT * from EMPLOYEE;

```
- Output:**

1 row(s) deleted.

1 row(s) deleted.
- Table:**

EMPLOYEEID	EMPLOYEENAME	EMPLOYEESSAL	EMPLOYEECITY	EMPLOYEEDOB
2	Rhea	6000	Brampton	15-JAN-99
3	Fahad	4000	Ajax	20-MAR-98
5	Akhilesh	9000	Oshawa	13-FEB-96

➤ Drop the Table:

The screenshot shows the 'Live SQL' interface. The top bar includes a 'Feedback' button, a 'Help' link, and a user email 'fahadismail053@gmail.com'. The main area has a dark header 'SQL Worksheet' with tabs for 'Actions', 'Save', and 'Run'. On the left, a sidebar lists navigation options: Home, SQL Worksheet (selected), My Session, Schema, Quick SQL, My Scripts, My Tutorials, and Code Library. The central workspace contains a code editor with the following content:

```
1 DROP TABLE EMPLOYEE;
```

Below the code, the message 'Table dropped.' is displayed.



## CONCLUSION

Time to wrap it up. What is your conclusion? How would you synthesize all the information into something even the busiest CEO wants to read? What are the key takeaways? How does your product/service/methodology uniquely address the issues raised by your study?

## Key Takeaways

### 1. Better Communication

A student database management system allows for easier communication between students, parents-teachers, students-teachers and even allows for the school alumni to stay connected with each other.

Instead of only relying on the school diary, the teacher can send instant updates to the parents related to school events, attendance, and even disciplinary issues over email or SMS. Students can connect with each other and with their teachers over on the discussion boards to clear their doubts and collaborate over assignments.

### 2. Simplified Admission Process

The admission season can be a chaotic time of the year for every school, but with the aid of the student database management system, the process gets simplified for everyone involved. From managing applicant database online to accepting registrations etc. the prospective students can use the cloud-based system to submit the forms, upload essential documents, apply for scholarships, etc.

The online system also makes it easy for the school to send students updates on admission tests, interview dates, reminders as well as the final admission status. This makes the admission process faster, error-free and more secure.

### 3. Faster Attendance Management

The attendance management system uses a biometric system or an access card to maintain a quick and accurate record of the student and school staff attendance. The system saves class teachers tedious minutes every morning calling out the class attendance and provides with the administration team on timely summaries and records of regular latecomers, absentees etc. when needed.

The system also allows for attendance updates to be sent to parents. It enables schools to track the attendance of the teachers and the various staff and calculate leaves and working days with ease and accuracy.

