DDL Questions

1. **Create a new database** called SchoolDB.

2. **Show all available databases** to verify that SchoolDB has been created.

3. Switch to the SchoolDB database.

4. **Create a table** called Students with the following structure:

• student\_id (INT) - Primary key

• first\_name (VARCHAR(50))

• last\_name (VARCHAR(50))

• grade (VARCHAR(5))

• age (INT)

create table practice.Students (

student\_id INT Primary key,

first\_name VARCHAR(50) ,

last\_name VARCHAR(50) ,

grade VARCHAR(5) ,

age INT)

5. **Create another table** called Courses with the following structure:

• course\_id (INT) - Primary key

• course\_name (VARCHAR(100))

• teacher (VARCHAR(50))

create table practice.Courses (

course\_id INT Primary key,

course\_name VARCHAR(100),

teacher VARCHAR(50) )

**6. Modifying the Tables:**

**Add a column** email to the Students table.

**Change the data type** of the age column in the Students table to TINYINT.

alter table schooldb.students

modify age tinyint

**7. Renaming and Dropping Tables**

**A. Rename the Courses table to Subjects**

alter table courses rename subjects

1. B. **Delete the** Subjects **table** permanently from the database
2. drop table schooldb.subjects

8. **Removing Data from the Tables**

**Truncate the** Students **table** to remove all records but keep the table structure intact.

truncate table schooldb.students

9. **Drop the** SchoolDB **database** after completing the assignment.

drop schema schooldb

**10. Creating a Table with Constraints**

1.

**Create a table** called Teachers with the following structure:

⚫ teacher\_id (INT) - Primary key

⚫ first\_name (VARCHAR(50))

⚫ last\_name (VARCHAR(50))

⚫ email (VARCHAR(100)) - Must be unique

⚫ hire\_date (DATE) - Cannot be NULL

CREATE TABLE `practice`.`teachers` (

`teacher\_id` INT NOT NULL,

`first\_name` VARCHAR(50) NULL,

`last\_name` VARCHAR(50) NULL,

`email` VARCHAR(100) NULL,

`hire` DATE NOT NULL,

UNIQUE INDEX `email\_UNIQUE` (`email` ASC) VISIBLE,

PRIMARY KEY (`teacher\_id`));

**11. Creating a Table with a Foreign Key:**

**Create a table** called Enrollments that tracks which students are enrolled in which courses. The structure should be:

1. enrollment\_id (INT) - Primary key

2. student\_id (INT) - Foreign key referencing Students(student\_id)

3. course\_id (INT) - Foreign key referencing Courses(course\_id)

4. enrollment\_date (DATE)

create table practice.Enrollments (

enrollment\_id int primary key,

student\_id int,

course\_id int,

enrollment\_date date,

foreign key (student\_id) references practice.students(student\_id),

foreign key (course\_id) references practice.courses(course\_id)

)

**12. Dropping a Column from a Table**

**Drop the** email **column** from the Students table.

alter table practice.students

drop column email

**13. Adding a Default Value to a Column**

**Add a** status **column** to the Students table with a default value of 'Active'. This column will track whether a student is currently active.

alter table practice.students

add column status varchar(50) default 'Active';

**14. Creating an Index**

**Create an index** on the last\_name column in the Students table to improve query performance when searching by last name.

create index last\_name\_index on practice.students(last\_name)

15. **Rebuild an index** that you created in Task 10 to improve its performance after data changes.

ALTER TABLE practice.teachers DROP INDEX last\_name\_index, ADD INDEX idx\_last\_name(last\_name);

**Advanced Level SQL DDL Assignment Questions**

**Partitioning a Table**: Create a Students table with partitioning based on the grade column, where students in different grades are stored in separate partitions.

alter table practice.students

partition by range (grade) (

partition low\_grade values less than (5),

partition mid\_grade values less than (7),

partition high\_grade values less than maxvalue

)

**Creating a View**: Create a view called ActiveStudentsView that displays the student\_id, first\_name, and last\_name of all students who have a status of 'Active' from the Students table.

**Using Check Constraints**: Create a Grades table with the following columns: student\_id, course\_id, and grade. Add a CHECK constraint to ensure that the grade value is between 0 and 100.

**Creating a Composite Primary Key**: Create a table called ClassroomAssignments with a composite primary key consisting of the teacher\_id and classroom\_id.

**Adding and Removing Foreign Key Constraints**: Add a foreign key constraint to the Enrollments table that references the teacher\_id column in the Teachers table. Then, remove that foreign key constraint.

**Using Auto-Increment**: Create a LibraryBooks table where the book\_id is automatically incremented whenever a new book is added to the library.

**Cloning an Existing Table Structure**: Create a new table called ArchivedStudents by cloning the structure of the existing Students table without copying the data.

**Creating a Trigger**: Write a trigger that automatically updates the status column in the Students table to 'Inactive' when the enrollment\_date is older than 4 years in the Enrollments table.

**Defining a Unique Constraint on Multiple Columns**: In the Enrollments table, add a UNIQUE constraint on the combination of student\_id and course\_id to ensure a student cannot enroll in the same course more than once.

**Renaming a Column**: Rename the first\_name column in the Students table to given\_name without losing the existing data.