**Name - Alisha Khatoon**

**Roll – 2023UGCS066**

**DBMS ASSIGNMENT- 8 :**

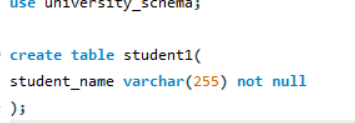
**Intermediate SQL Topics**

Exercise on Integrity Constraints

1. Create a table with a NOT NULL constraint on the "name" column in the student table.

use universityschema;

create table student1( student\_name varchar(255) not null );



***OUTPUT:***



2. Add a UNIQUE constraint on the "name" column in an existing student table.

alter table student

add constraint unique\_name unique(name);



3. Create a Employee table with a CHECK constraint ensuring age is at least 18.

create table employee

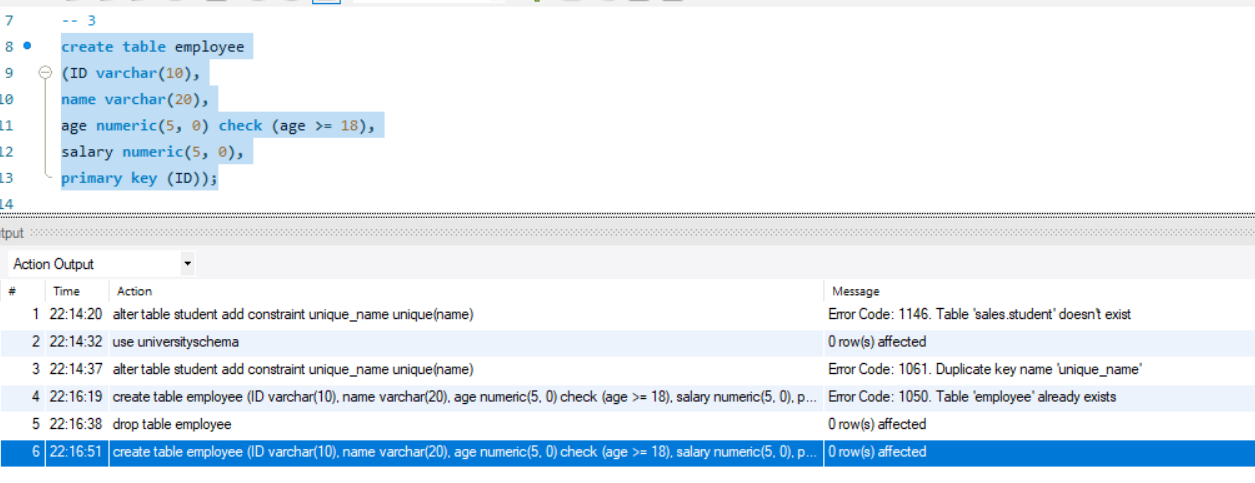
(ID varchar(10),

name varchar(20),

age numeric(5, 0) check (age >= 18),

salary numeric(5, 0),

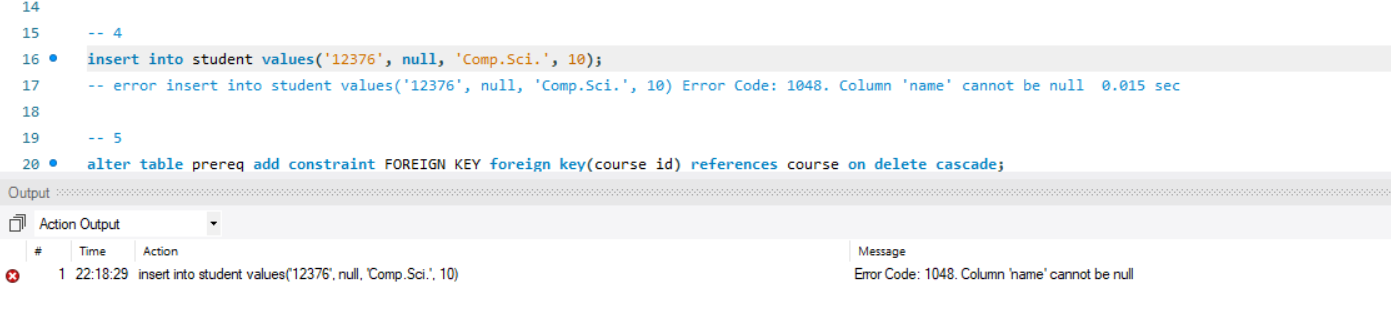
primary key (ID));



4. Insert a record violating a NOT NULL constraint and observe the error.

insert into student values('12376', null, 'Comp.Sci.', 10);

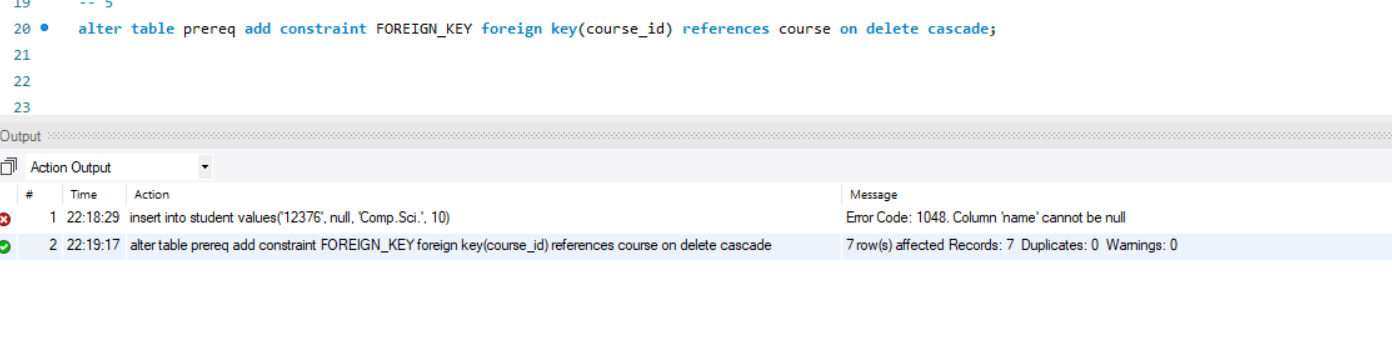
-- error insert into student values('12376', null, 'Comp.Sci.', 10) Error Code: 1048. Column 'name' cannot be null 0.015 sec



5. Define a FOREIGN KEY constraint linking "course\_id" in "prereq" to "course\_id" in "courses".

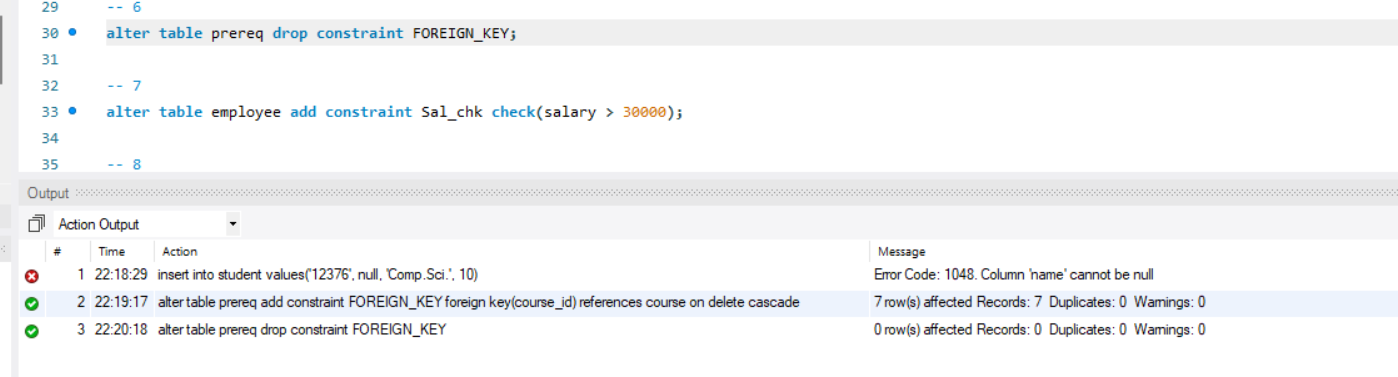
alter table prereq add constraint FOREIGN\_KEY foreign key(course\_id) references course on delete cascade;

-- insert into prereq values('45379', '78906');



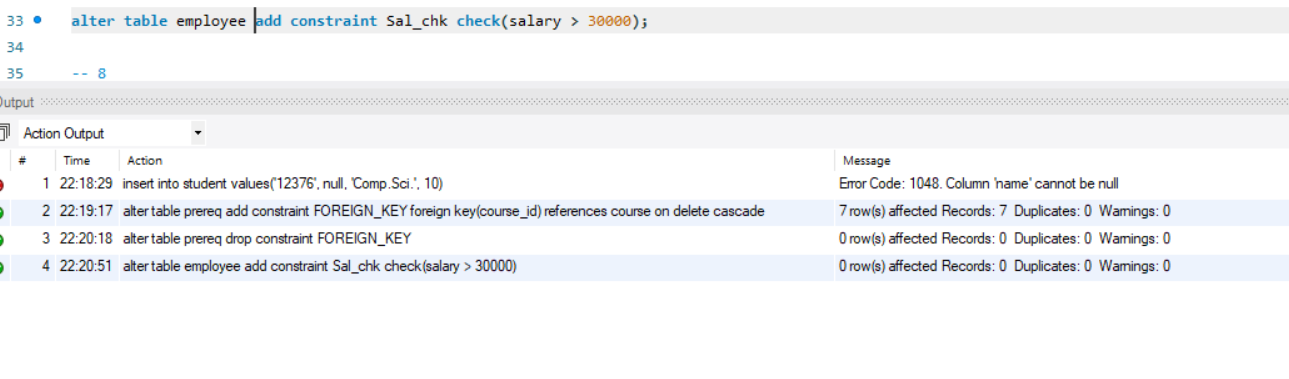
6. Drop a FOREIGN KEY constraint from the "prereq" table.

alter table prereq drop constraint FOREIGN\_KEY;



7. Modify an existing table to add a CHECK constraint for salary > 30000.

alter table employee add constraint Sal\_chk check(salary > 30000);



8. Demonstrate the ON DELETE CASCADE action in FOREIGN KEY constraints.

select \* from student;

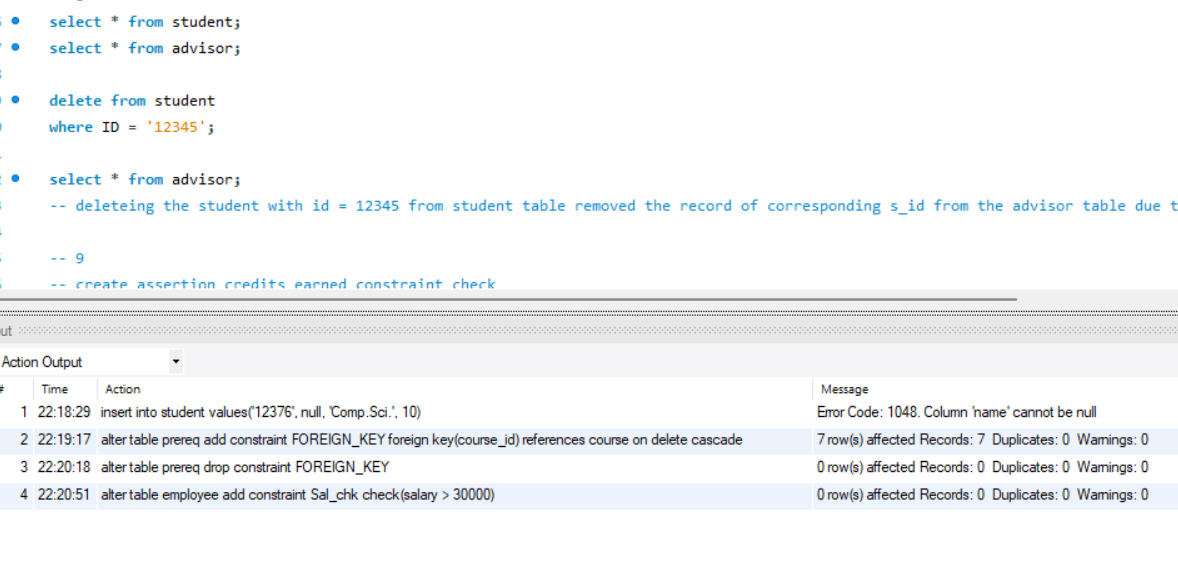
select \* from advisor;

delete from student

where ID = '12345';

select \* from advisor;

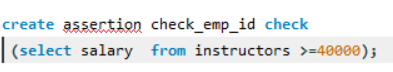
-- deleteing the student with id = 12345 from student table removed the record of corresponding s\_id from the advisor table due to freign key constraint with on delete cascade.



9. Use the ASSERTION feature to enforce a constraint across multiple tables (if supported).

create assertion check\_emp\_id check (

select salary from instructors >=40000);



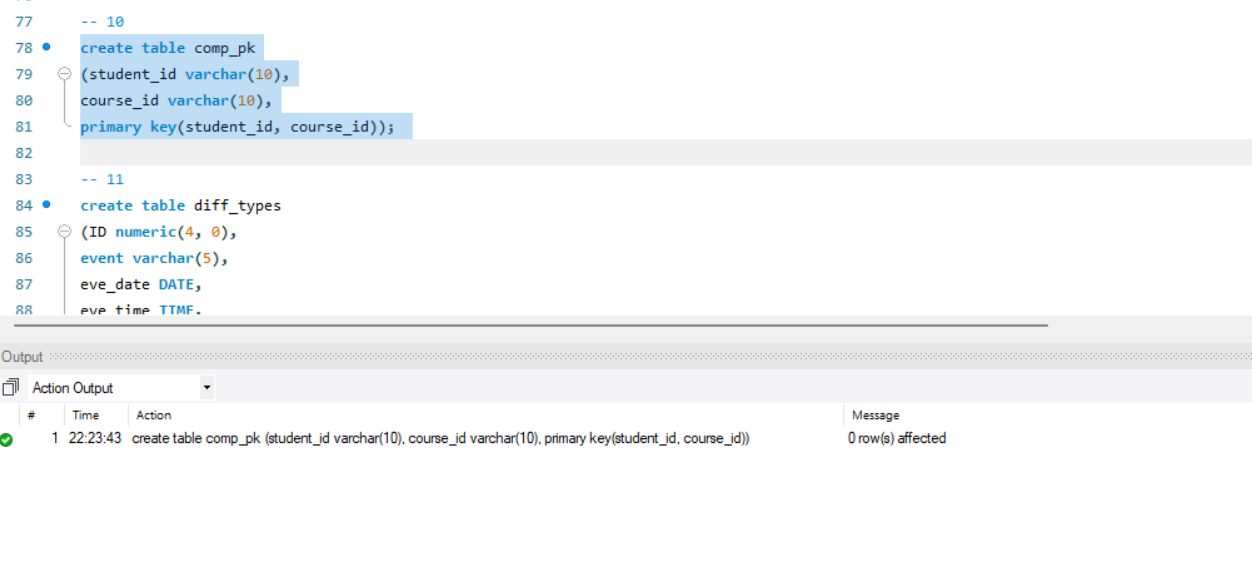
10. Create a table with a composite primary key (student\_id, course\_id).

create table comp\_pk

(student\_id varchar(10),

course\_id varchar(10),

primary key(student\_id, course\_id));



Exercise on SQL Data Types and Schemas

11. Create a table using different SQL data types including DATE and TIME.

create table diff\_types

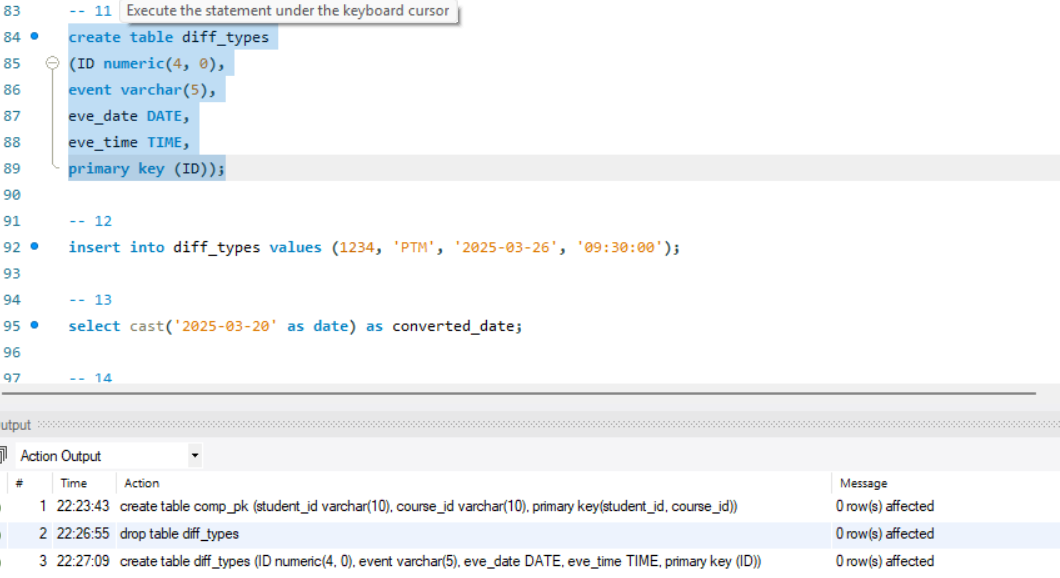
(ID numeric(4, 0),

event varchar(5),

eve\_date DATE,

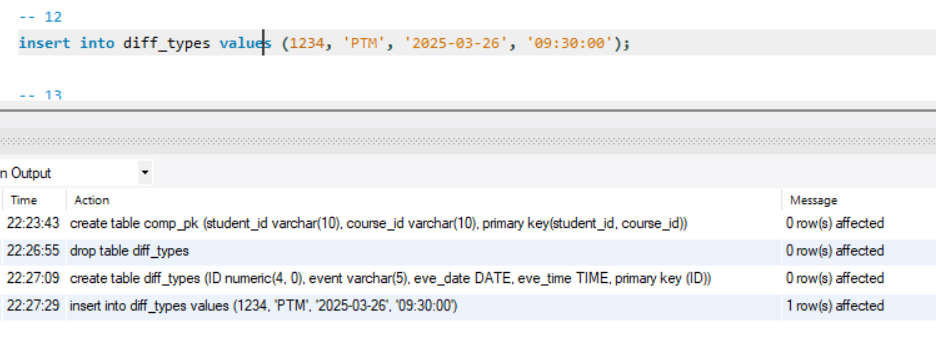
eve\_time TIME,

primary key (ID));



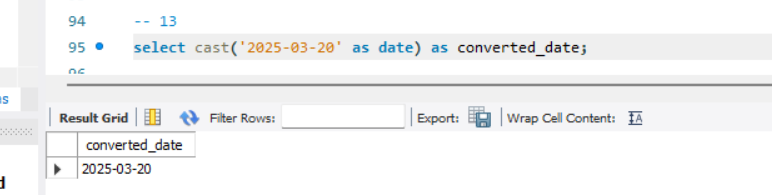
12. Insert a record into a table with a DATE and TIME column.

insert into diff\_types values (1234, 'PTM', '2025-03-26', '09:30:00');



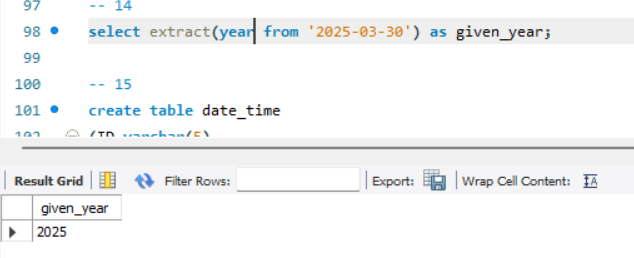
13. Demonstrate the use of CAST to convert a string to a DATE type.

select cast('2025-03-20' as date) as converted\_date;



14. Use EXTRACT to retrieve the year from a DATE column.

select extract(year from '2025-03-30') as given\_year;



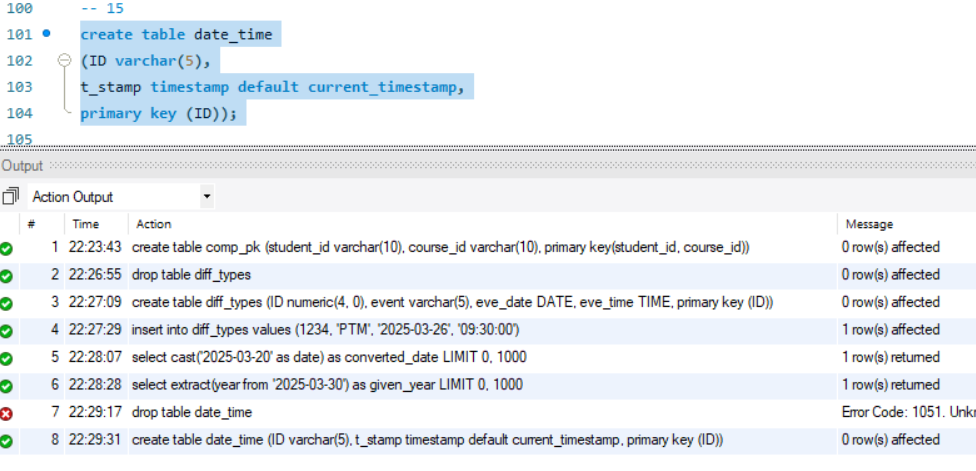
15. Create a table with a DEFAULT constraint for a timestamp column.

create table date\_time

(ID varchar(5),

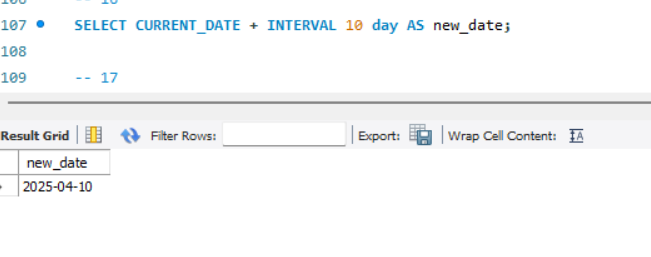
t\_stamp timestamp default current\_timestamp,

primary key (ID));



16. Demonstrate the usage of the INTERVAL type in date calculations.

SELECT CURRENT\_DATE + INTERVAL 10 day AS new\_date;



17. Showcase the use of ENUM data type (if supported) for predefined values.

create table section

(course\_id varchar (8),

sec\_id varchar (8),

semester ENUM('Fall', 'Winter', 'Spring', 'Summer'),

year numeric (4,0) check (year > 1701 and year < 2100),

building varchar (15),

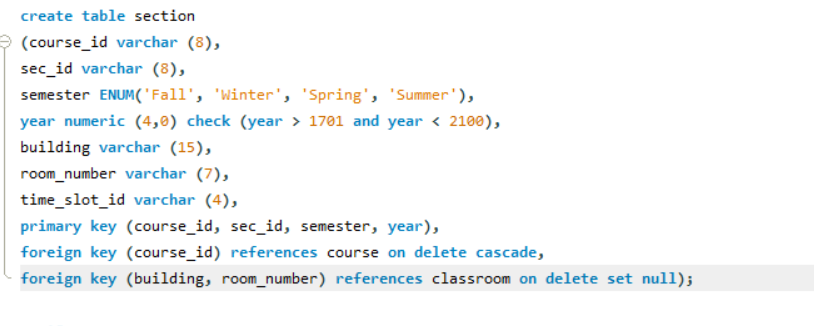
room\_number varchar (7),

time\_slot\_id varchar (4),

primary key (course\_id, sec\_id, semester, year),

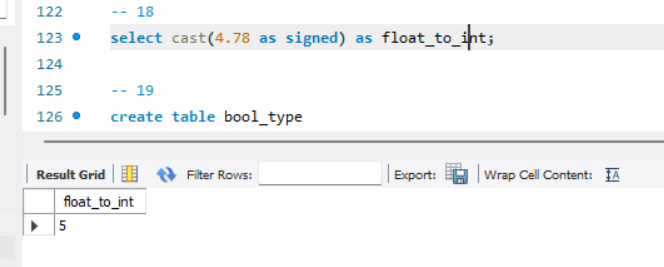
foreign key (course\_id) references course on delete cascade,

foreign key (building, room\_number) references classroom on delete set null);



18. Convert a FLOAT value to an INTEGER using CAST.

select cast(4.78 as signed) as float\_to\_int;



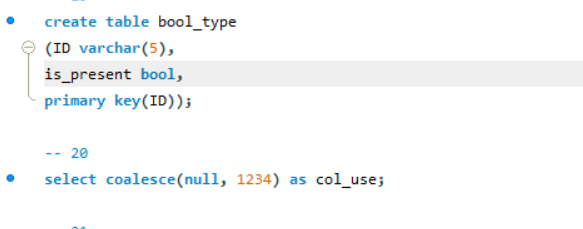
19. Create a table with a BOOLEAN data type (if supported).

create table bool\_type

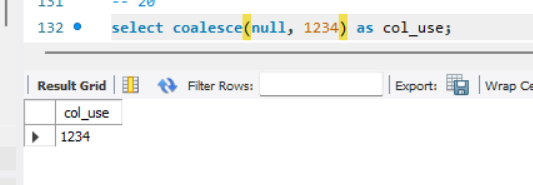
(ID varchar(5),

is\_present bool,

primary key(ID));



20. Demonstrate the COALESCE function to handle NULL values select coalesce(null, 1234) as col\_use;



Exercise on Index Definition in SQL

21. Create an index on the "name" column of the "student" table.

CREATE INDEX idx\_student\_name ON student(name);



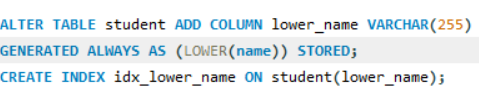
22. Create a function-based index (if supported).

ALTER TABLE student ADD COLUMN lower\_name

VARCHAR(255) GENERATED ALWAYS AS

(LOWER(name)) STORED;

CREATE INDEX idx\_lower\_name ON student(lower\_name);



23. Create a UNIQUE index on the "name" column of the "students" table.

CREATE UNIQUE INDEX unique\_idx\_student\_name ON student(name);



24. Drop an existing index named "idx\_student\_name".

DROP INDEX idx\_student\_name ON student;



25. Use the EXPLAIN command to analyze query performance with and without an index.

EXPLAIN ANALYZE SELECT \* FROM student WHERE name = 'snow';



***OUTPUT:***



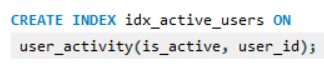
26. Create a composite index on "first\_name" and "last\_name".

CREATE INDEX idx\_name ON student(first\_name, last\_name);



27. Create a partial index to store only active users.

CREATE INDEX idx\_active\_users ON user\_activity(is\_active, user\_id);



28. Measure query performance before and after adding an index.

EXPLAIN ANALYZE SELECT \* FROM student WHERE name = 'Tanaka';



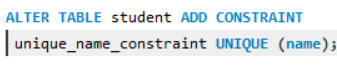
***OUTPUT:***



29. Use a UNIQUE constraint as an alternative to a unique index.

ALTER TABLE student ADD CONSTRAINT

unique\_name\_constraint UNIQUE (name);



30. Create a FULLTEXT index for faster text searching (if supported).

CREATE FULLTEXT INDEX idx\_fulltext ON student(name);



Exercise on Authorization

31. Grant SELECT privilege on the "student" table to user "user1".

GRANT SELECT ON student TO user1;



32. Revoke INSERT privilege on the "student" table from "user1".

REVOKE INSERT ON student FROM user1;



33. Grant ALL privileges on the "department" table to "admin".

GRANT ALL ON department TO admin;



34. Grant SELECT and UPDATE privileges to multiple users.

GRANT SELECT, UPDATE ON student TO user1, user2;



35. Create a new role and assign privileges to it.

CREATE ROLE manager\_role;

GRANT SELECT, UPDATE ON employee TO manager\_role;



36. Check the privileges granted to a user.

SHOW GRANTS FOR user1;

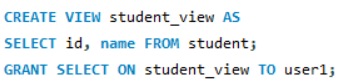


37. Create a view with restricted access for certain users.

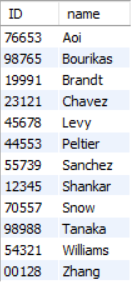
CREATE VIEW student\_view AS

SELECT id, name FROM student;

GRANT SELECT ON student\_view TO user1;



***OUTPUT:***



38. Use WITH GRANT OPTION to allow a user to grant privileges to others.

GRANT SELECT ON student TO user1 WITH GRANT OPTION;



39. Assign the "manager" role to a user.

GRANT manager\_role TO user3;



40. Revoke DELETE privilege from the "manager" role.

REVOKE DELETE ON employee FROM manager\_role;

