DATABASE MANAGEMENT SYSTEMS – LABORATORY

AY: 2025-26 Sem – I

**Name of the Student**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Class:** TE **Division**: \_\_\_\_\_\_ **Batch:** \_\_\_\_\_\_\_\_ Roll **No**: \_\_\_\_\_\_\_\_\_\_

**Assignment No: 2**

**Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence and Synonym**

* DDL Commands: CREATE, DROP, ALTER, RENAME, TRUNCATE
* DML Commands: SELECT, INSERT, UPDATE, DELETE

For every command / sub question below:

* Paste screenshots of the commands and the executed queries
* In case of typical cases, demonstrate the errors
* Ensure use of your name and roll no for the databases
* Ensure your name appended with your last name for the creation of the tables

|  |  |
| --- | --- |
|  | **Difference in Mysql workbench , Mysql line – command, sudo mysql**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | **Show Databases**  Also try commands like   * Show database * SHOw DaTaBaSe   Derive Conclusions on the case sensitivity in mysql  Try to create the databases by using - pseudo mysql and client server option …….. check for the error and draw conclusion on the “database administrator and analyst”  *(Ensure use of your name and roll no for the databases)* |
|  | **Create Database**  *(Ensure creation of a database with your name)*   * Check if you can create a database when logged in using pseudo mysql ? justify the results * After creation of the database, again use to command ‘show databases’ to see it. |
|  | **Show Tables**    *(Ensure use of your name appended with Last name for the table name)* |
|  | **Create Table**   * Try to assign a name to an attribute with space in between ex: Roll No , First name. check the error. * Execute the command “show tables: to check to created table |
|  | **Describe and Desc**  Create a table named student and describe it   * Desc student1 * Describe student1 * Desc table student1 * Describe table student1   Derive conclusion on Desc table / Describe table commands. Paste the screen shots to demonstrate the outcomes in both the cases |
|  | **Show Create table**   * Show create table student1 * Compare the results of all three commands   – desc,   * describe, * show create |
|  | **Insert commands**   * Only one value at a time - in a single command * Multiple values in a single command * Selective values (only some of the defined attributes)   *(Insert your name other details and* ***five students after your roll no****)* |
|  | **Display Table (Select \*from)**   * Compare the outcomes with ‘Show create table’ student1 |
|  | **Drop**   * Implement an example of drop * Again, use show tables to verify |
|  | **Truncate**   * Implement an example of truncate and demonstrate the difference in drop & truncate   *(Paste screenshots of the commands and the executed queries / errors)* |
|  | **Delete**   * Mention an example of delete and demonstrate the difference in Delete, Drop & Truncate   *(Paste screenshots of the commands and the executed queries / errors)* |
|  | **ALTER**   * Create Faculty table * Faculty {E\_no, F\_name, L\_name, Gender, Date of Joining, Course\_Id} * Write a query on above table by using ALTER with * ADD, * MODIFY, * DROP * RENAME   *(Paste screenshots of the commands and the executed queries / errors)*   * **Alter – Add:** - Add a column to a table (Check where the column is added by default) * **Alter – Add: First | After column\_name -** specify the position of the new column in the table. (Add the column before / after the existing column) * **Alter - Add:** Add multiple columns in a single command * **Alter – Modify:** Specify datatype and length – ex: varchar (25) * **Alter – Modify** – specify datatype and not length datatype –int -small int, * **Alter – Modify** – Null to not-null, not null is null etc (if a few attributes are already null and then if you are trying to modify it to null, mark the error that you get) * **Alter – Modify - Decimal (6,2):** write the significance of the command decimal (6,2) * **Alter – Drop** * **Alter – Change / Rename** |
|  | **NULL Constraints**   * Create a Table without any NULL Constraints. * Describe the table * Alter the table to include Null Constraints * - Make Roll\_no Not Null * - Make Name field Not null * Describe the table to verify it * Use ALTER Command * “Alter table student1 modify roll\_no int not null” * Define an attribute permitted to be Null and then after leaving some un-inserted values try to now apply Null Constraint. Is it permitted? * Draw conclusions after inserting a few values in the tables:   changing Null Not Null  changing Not Null Null |
|  | **Unique Constraints**   * Already a table has been created and now alter it to add constraint * Initialize unique constraint while creating table * Make one of the fields-say – Roll\_no as a unique key * Try to insert duplicate values * Describe the table to verify the application of unique constraint on an attribute * Drop the defined unique constraints * Use alter to again define it.   Try multiple cases / combinations in the defined unique constraints:   * Case 1: Only one attribute – Roll\_no is defined as an unique constraint * Case 2: Roll\_no was already defined as a unique constraint earlier. But now a combination of one more field is used along with roll\_no: Roll\_no + f\_name * Case 3: Roll\_no was already defined as a unique constraint earlier. But now a combination of two more fields is used along with roll\_no : Roll\_no + age + l\_name * Case 4: Earlier ‘Roll\_no’ was ‘unique’ key and was used in the combination. Now the combination of two fields does not include the unique key. The combination is of ‘age and F-name’. The desc command describes the key age as multivalued. * Case 5: Earlier ‘Roll\_no’ was ‘unique’ key and was used in the combination. Now the combination of three fields does not include the unique key. The combination is of ‘l\_name, age, F-name’. The desc command, describes the keys: L-name and age as multivalued.   After each case, use describe command to check the attribute which is defined as multivalued – MUL  *(Paste screenshots of the commands and the executed queries / errors)* |
|  | **Primary Constraint**   * Already a table has been created and now alter it to add primary constraint * Initialize unique constraint while creating table * Make one of the fields-say – Roll\_no as a primary key * Describe the table to verify the application of primary constraint on an attribute * Use a combination of attributes and define them as primary constraint. * Describe command – verify the defined primary constraints – single and multiple * Drop the defined primary constraints * Replace the already defined primary constraints * Use alter to again define it. |
|  | **Primary Key and Auto Increment**   * You have created a table where say, the ‘roll\_no’ is the primary key and it is autoincrement * Now, if you try to delete a primary key (which is auto increment), you will get an error. * Alter Table – define autoincrement attribute as simple int and then drop primary key   Derive the conclusions about primary key and autoincrement |
|  | **Foreign Key**   * Create Parent Table * Create Student Table * Try to insert the values which are not present in the parent table – error (Referential integrity) * Include examples of the following (with errors initially and then without error)   – on-delete cascade and on-update cascade   * on-delete set null and on-update cascade |
|  | **Check Constraint**   * Single condition * Multiple condition * Alter * Delete |
|  | **Sort / Order By**   * Descending * Ascending * If nothing – Desc / Asc is mentioned, explore the default trend * Mix- Sort by multiple columns Ex: Order by Asc, Salary desc * Sort Alphabetically |
|  | **Select Distinct**  Say you have a table – Student with roll\_no, Name and Department   * Only one distinct attribute:   Ex: Select Distinct Departments From Student   * Couple of distinct attributes:   Ex: Select Distinct Departments, Name From Student   * Count of distinct attribute   Ex: Select Count Distinct departments from student   * Ordered Distinct Attribute:   Select Distinct Departments from student order by departments Desc |
|  | **Index**   * Search the f\_name (first tuple) from student table and use explain syntax * Search the S\_ID (Last tuple) from student table and use explain syntax * Comment on above (clustered index and non-clustered index) * Create index on f\_name * Search the f\_name from student table and use explain syntax * Comment on above * Show clustered and non-clustered index |
|  | **Create View Table:**   * Create view using only one base table and without a where clause * Create view using only one base table and with a where clause * Create view using multiple base-tables without matching where clause (Analyze the correct but unexpected result – cartesian product) * Update view (created using a single base table) and show the sudation in the corresponding base table (updatable View concept) * Update the base table and see the corresponding updation in the created - view – table (updatable View concept) * Update the view created using multiple base tables – check if you get an error * use the command show tables * Draw conclusions on the tables seen (are virtual tables included? ) |