|  |  |  |  |
| --- | --- | --- | --- |
| No. | Title | Points to be covered / example | Length |
| 1 | Introduction to data types. | Note: Welcome to the session on Introduction to data types which is basic but very important topic to design databases.  What we going to cover in this session/ In this session we are going to cover/  Today’s topic are / Today we are going to discuss/.  Categories of datatype Numeric, string, binary  Discussion on numeric data types (int, smallint, etc) including their ranges.  Difference between dec and float.  Discussion on String datatype (char, varchar, text, etc) non Unicode  Diff between char and varchar  Discussion on nvarchar/ Unicode  Diff between varchar and nvarchar (2)  Discussion on BLOB (full form to be told) – small, bigblob, etc.  Tell them the simillraty between blob and string.  CHAR, VARCHAR, TINYINT, INT, YEAR, DATE, BOOLEAN  Char(10)  Varchar(10)  e.g. todo  e.g.  DECIMAL(7, 2) Total length of this data type is 7 of which 2 will be decimal part.  5396.17 ok value  5396.174 5396.17  5396.178 5396.18  5396.1743 5396.17  5396.1748 5396.17  5396.1788 5396.18  Note: With this we have covered all the today’s topics.  I hope you understood data types in SQL. | 15 |
| 2(P1) | SQL Commands | Note: Welcome back to the session on SQL Commands.  What are SQL commands   1. DDL – Data Definition Language 2. DML – Data Manipulation Language 3. DCL – Data Control Language 4. TCL – Transaction Control Language 5. Constraints   DDL – Discuss – Full list of commands with brief description  e.g. of DDL  **CREATE** TABLE student (  ID INT,  firstName VARCHAR(45),  lastName VARCHAR(45),  DoB DATE  );  **ALTER** TABLE student ADD COLUMN emailID VARCHAR(145);  **DROP** TABLE student;  DML– Discuss – Full list of commands with brief description  e.g. of DML  **INSERT** INTO student VALUES (1, 'raju', 'patel', '1970-12-10', 'raju123@gmail.com');  **SELECT** \* FROM student;  **UPDATE** student SET emailID = 'mohan.desai@gmail.com' WHERE ID = 3;  **DELETE** FROM student WHERE ID = 3;  DCL– Discuss – Full list of commands with brief description  **GRANT and REVOKE**  TCL– Discuss – Full list of commands with brief description  **COMMIT and ROLLBACK**  Discussion on types Of Constraints (in brief)   * **NOT NULL** - Ensures that a column cannot have a NULL value * **UNIQUE** - Ensures that all values in a column are different * **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table * **FOREIGN KEY** - Uniquely identifies a row/record in another table. * **CHECK** - Ensures that all values in a column satisfy a specific condition. | 20 |
| 3(P1) | Database management system | Intro to this course (Ulka and saleel - before recording discuss with sir)  Pre req. – For whom this course imp (eg. Application dev, DBA’s, Data Mining  Career imp - to become DBA, Data Sciten, etc.  Why this course is easy to learn (Section of course, number of sessions, total duration course duration 30hrs.)  Examples been covered. (number of databases, tables discuses)  Note: To be record at the last, content continually updated. | 5 |
| 4(P3) | Introduction to RDBMS | TODO: | 8 |
| 5(P3) | File System vs DBMS | Concepts of Flat Files  Concepts of DBMS  Advantage of File-oriented system  Backup, Data retrieval, Editing, Remoteaccess, Sharing  Disadvantage of File-oriented system  Data redundancy, Datainconsistency, Limiteddatasharing, Datasecurity  Advantage of DBMS  Improveddatasharing, Improveddatasecurity, Minimizeddatainconsistency, Minimized dataredundancy  Disadvantage of DBMS   * Cost of Hardware and Software of a DBMS is quite high * Most database management systems are often complex systems, so the training for users to use the DBMS is required. * All data is integrated into a single database which can be damaged because of electric failure or database is corrupted on the storage device.   Slides with graphics explaining the above concepts (good explanatory images)  Note: Content continually updated. | 5 |
| 6(P1) | Schema in databases | Intro to schema  A **schema** is a collection of database objects including **tables**, **views**, **triggers**, **stored** **procedures**, **indexes**, etc. A **schema** is associated with a username which is known as the **schema** owner, who is the owner of the logically related database objects.  A database schema can be divided broadly into two categories −   * **Physical Database Schema** − This schema is related to the actual storage of data and its form of storage like files. It defines how the data will be stored in a secondary storage. * **Logical Database Schema** − This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.   Defining and creating schema  To create a new database in MySQL, you use the CREATE DATABASE statement with the following syntax:  CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] *db\_name*  e.g.   * CREATE SCHEMA IF NOT EXISTS myDB; * CREATE DATABASE IF NOT EXISTS myDB;   To select a particular database to work with you issue the USE statement with the follows syntax :  USE *db\_name*  The USE statement tells MySQL to use the named database as the default (current) database for subsequent statements. The named database remains the default until the end of the session or another USE statement is issued.  e.g.   * USE myDB;   Managing and moving objects between schemas  Note: Moving objects from one schema to another schema in MySQL is not possible  Note: Slides required, live examples/demos ( Create Schema/database in MySQL is same, Use schema, moving objects (in MS-SQL possible)from one schema to another schema) | 10 |
| 7(P1) | DDL Commands | **DDL** (Data Definition Language) statements or commands are used to define and modify the database structure of your tables or schema.  **CREATE** TABLE statement is used to specify the layout of your table  The table creation command requires the following details.   * Name of the table * Name of the fields * Definitions for each field   syntax to create a MySQL table is as follows  CREATE TABLE table\_name (column\_name column\_type);  CREATE TABLE student (  ID INT(11) PRIMARY KEY,  firstName VARCHAR(45),  lastName VARCHAR(45),  DoB DATE  );  **ALTER** command is very useful when you want to change a name of any table field or if you want to add or delete an existing column in a table.   * ALTER TABLE student ADD COLUMN emailID VARCHAR(128); * ALTER TABLE student CHANGE ID studentID INT; * ALTER TABLE student DROP COLUMN emailID;   DROP command, it is very easy to drop an existing MySQL table, but you need to be very careful while deleting any existing table because the data lost will not be recovered after deleting a table.   * DROP TABLE student;   Disc on CREATE, ALTER, DROP  Disc this commands with ref to tables |  |
| 8(P1) | Candidate and Primary key | What is a candidate key?  **Candidate Key** – A Candidate Key can be any column or a combination of columns that can qualify as unique key in database. There can be multiple Candidate Keys in one table. Each Candidate Key can qualify as Primary Key.  What is a primary key?  **Primary Key**: Primary Key is a set of attributes (or attribute) which uniquely identify the tuples in relation or table. There is one and only one primary key in any relationship.  Here in **student** table you can choose either **ID**, **pan\_card**, or **emailID** columns as primary key column, here **ID** cab be a preferable choice.  Diff between CK and PK   |  |  | | --- | --- | | **Primary Key** | **Candidate Key** | | There can be only one primary key in any relation. | There can be more than one candidate key in a relation. | | Primary key can not contain NULL value. | Candidate key can have NULL value. | | A primary key is a candidate key. | It not compulsory that each candidate key can be a primary key. | | Primary key specifies the important attribute for the relation. | Candidate specifies the key which can qualify for primary key. |   Points to consider to make column PK and CK | 10 |
| 9(P1) | Primary Key | Imp of PK  A primary key is one of the most important steps in good database design. A primary key is a special column (or set of combined columns) in a relational database table, that is used to uniquely identify each record. Each database table needs a primary key.  PRIMARY key must follow some rules.   * A primary key cannot be NULL. * A primary key value must be unique. * A table has only one primary key.   Examples of PK  CREATE TABLE student (  ID INT(11) PRIMARY KEY,  firstName VARCHAR(45),  lastName VARCHAR(45),  DoB DATE,  emailID VARCHAR(145)  );  PK and AUTO\_INCREMENT  CREATE TABLE student (  ID INT(11) PRIMARY KEY AUTO\_INCREMENT,  firstName VARCHAR(45),  lastName VARCHAR(45),  DoB DATE,  emailID VARCHAR(145)  );  PK data type mostly it must be INT(eg ID), (Eg email varchar PK)  Using multiple columns as PK (Composite Key)  **Composite key**, or **composite primary key**, refers to cases where more than one column is used to specify the **primary key** of a table. In such cases, all foreign **keys** will also need to include all the columns in the **composite key**.  Note that the columns that make up a **composite key** can be of different data types.  CREATE TABLE student (  ID INT(11),  firstName VARCHAR(45),  lastName VARCHAR(45),  DoB DATE,  emailID VARCHAR(145),  PRIMARY KEY (ID, emailID)  );  Note: With Slides and Examples (student Table R= {ID, firstName, lastName, DoB, emailID } ) | 15 |
| 10(P1) | Foreign Key | Concepts of FK  Examples of FK  Data type of PK/FK must be same  Note: With Slides and Examples (student Table R= {ID(PK), firstName, lastName, DoB, emailID } , student\_address Table, student\_hobbies table}) – insert 4-5 demo records  Structure view and Brows view  URL: <https://dev.mysql.com/doc/workbench/en/wb-creating-eer-diagram.html> |  |
| 11(P1) | Introduction to DML commands | **DML** stands for Data Manipulation Language. It is a language used for **selecting**, **inserting**, **updating** and **deleting** data in a tables.  Use of INSERT, SELECT, UPDATE, DELETE and TRUNCATE with single table (student table)  The **INSERT** **INTO** s**tatement** is used to add new rows of data to a table in the database.   * INSERT INTO student VALUES (1, 'ramesh', 'patel', '1999-10-17', 'ramesh.patel@gmail.com'); * INSERT INTO student VALUES (2, 'rajesh', 'mehta', '2000-12-20', 'rajesh.mehta@gmail.com'); * INSERT INTO student VALUES (3, 'vipul', 'shah', '2001-07-19', 'shahvipul@yahoomail.com'); * INSERT INTO student VALUES (4, 'kamlesh', 'kaka', '2002-11-26', 'kamlesh.kaka@gmail.com'); * INSERT INTO student VALUES (default, 'ramlal', 'kumar', '2000-11-07', 'ramlal447@gmail.com'); * INSERT INTO student VALUES (default, 'raj', 'sharma', '2001-12-20', 'raj1999@gmail.com'); * INSERT INTO student VALUES (default, 'bimal', 'verma', '1998-07-19', 'bimal1984@yahoomail.com'); * INSERT INTO student VALUES (default, 'kamlesh', 'kumar', '1999-11-26', 'kamlesh1623@gmail.com'); * INSERT INTO student (ID, firstName, lastName, DoB, emailID) VALUES (9, 'rajesh', 'mehta', '2000-12-20', 'rajesh.mehta@gmail.com'); * INSERT INTO student (firstName, lastName, DoB, emailID) VALUES ('raj', 'sharma', '2001-12-20', 'raj1999@gmail.com');   The **SELECT statement** is used to fetch the data from a database table which returns this data in the form of a result table.   * SELECT \* FROM student;   The **UPDATE statement** is used to **update** the data of an existing table in database. We can **update** single columns as well as multiple columns using **UPDATE statement** as per our requirement.   * UPDATE student SET email =NULL; * UPDATE student SET emailID = ' ramesh.patel123@yahoomail.com' where ID = 1;   The **DELETE Statement** is used to delete existing records from a table. We can delete a single record or multiple records depending on the condition.   * DELETE FROM student; * DELETE FROM student where ID = 1;   The TRUNCATE TABLE command deletes the data inside a table, but not the table itself.   * TRUNCATE TABLE student;   Use of above commands with related tables (PK/FK table – student - student\_address)   * INSERT INTO student\_address VALUES (1, 'paud road', 'pune'); * INSERT INTO student\_address VALUES (2, 'M.G. road', 'baroda'); * INSERT INTO student\_address VALUES (3, 'k.k road', 'surat'); * INSERT INTO student\_address VALUES (104, 'station road', 'baroda');   Referential integrity (Directly adding a record in student\_address, deleting a student record,  Similar for student\_hobbies table)   * INSERT INTO student\_hobbies VALUES (1, 1, 'running'); * INSERT INTO student\_hobbies VALUES (2, 1, 'reading'); * INSERT INTO student\_hobbies VALUES (3, 2, 'football'); * INSERT INTO student\_hobbies VALUES (4, 2, 'running'); * INSERT INTO student\_hobbies VALUES (105, 3, 'watching movies'); * INSERT INTO student\_hobbies VALUES (104, 4, 'gaming');   Note: With Slides and Examples (student Table R= {ID(PK), firstName, lastName, DoB, emailID } , student\_address Table, student\_hobbies table})  Structure view and Brows view | 20 |
| 12(P2) | Working with ER models | What are ER models  Working with MySQL ER model Tool.  Benefits of documentation  Third party ER database documentations tools (toad from DELL, NaviCat)  Note: With Slides and Examples | 5-10 |
| 13(P3) | Using ER model for Project Documentations | Attributes of ER models  TODO |  |
| 14(P1) | Introduction to Relationships | Basic intro to all relationships  Note: With Slides with relationship definition and ER images | 5 |
| 15(P1) | One-to-one Relationship | Discussion of one-to-one relationships  Design intents for creating one-to-one relationships (situations like student\_address, one more example to be added)  How to create one-to-one relationship (T1-PK/T2-PK&FK)  Note: With Slides and Examples | 10 |
| 16(P1) | One-to-many Relationship | Discussion of one-to-many relationships  Design intents for creating one-to-many relationships (situations like student\_hobbies, one more example of **invoice** and **invoice\_items**) (Every one invoice must have at least one item))  How to create one-to-many relationship  Note: With Slides and Examples | 10 |
| 17(P1) | Many-to-many Relationship | Discussion of many-to-many relationships  Design intents for creating many-to-many relationships (situations like student, course, and course\_student)  How to create many-to-many relationship  Note: With Slides and Examples | 10 |
| 18(P4) | Many-to-many Relationship - 2 | Solving advance problem statements | TODO |
| 19(P2) | Introduction to Normalization | Concepts of Normalization  Reasons to normalize the database.  Brief on all normalization levels  Note: With Slides-good graphics | 10 |
| 20(P2) | Normalization | Insertion, Updating, and Deletion Anomaly  Note: With Slides and example | 5-10 |
| 21(P2) | First form of Normalization | Def of first form of normalization (1NF)  Understanding of closure of functional dependency  Understanding functional dependency and its properties.  Note: With Slides-good graphics and example | 10 |
| 22(P2) | Functional Dependency | Disc on Function Dependency  Closure of FD  Properties of FD  Note: With Slides-good graphics and example |  |
| 23(P2) | Second Normalization | Dis on 2NF  Note: With Slides-good graphics and example |  |
| 24(P2) | Third Normalization Form | Dis on 3NF  Problem statement for 3NF. (on Faculty/Room/Timing)  Note: With Slides-good graphics and example |  |
| 25(P4) | Boyce Codd Normalization | Dis on BCNF |  |
| 26(P4) | Fifth Normalization Form | Dis on 5thNF  Loss less and loose decomposition  Minimal coverage  Note: With Slides-good graphics and example |  |
| 27(P2) | FAQ on Normalization | Recap on Normalization (1, 2 and 3NF)  15 Interview Que and Ans on Normalization.    Note: With Slides-good graphics and example |  |
| 28(P1) | Introduction to Joins | Dis on JOINS  The **SQL Joins** clause is **used** to combine records from two or more tables in a database. A **JOIN** is a means for combining fields from two tables by using values common to each.  Need for Joins  Types of Joins   * Cartesian or Product Join – Cross Join * Equijoin – Inner Join * Natural Join * Outer Join – Right Outer Join, Left Outer Join * Self-Join   Note: With Slides-good graphics and example | 10 |
| 29(P1) | Inner/ equi Joins | Dis on Inner/equi Joins  Dis on syntax  Situations to use Inner Joins.  Select appropriate tables (Student, student\_address)  Diff between Inner/equi and Natural join  Note: With Slides-good graphics and example | 10 |
| 30(P1) | Natural Join | Dis on Natural Join (Create new tables for Natural Join) keep the tables ready.  Diff between Natural join and Inner/equi Join.  Note: With Slides-good graphics and example | 5 |
| 31(P1) | Self Join | Dis on Self Join  Note: With Slides-good graphics and example |  |
| 32(P1) | Left Outer Join | Dis on Left Outer Join  Examples Student/CourseFees/Library  Note: With Slides-good graphics and example |  |
| 33(P1) | Right Outer Join | Dis on Right Outer Join  Examples Student/CourseFees/Library  Note: With Slides-good graphics and example |  |
| 34(P1) | Cross Join / Cartesian Join | Dic on Cross Join and Cartesian join    E.g. multi location chain of shops and their product inventory.  URL: <https://www.sqlservertutorial.net/sql-server-basics/sql-server-cross-join/>  Note: With Slides-good graphics and example |  |
| 35(P1) | UNION | Definition of UNION and UNION ALL  E.g. of union with diagram  URL: <https://www.sqlshack.com/sql-union-overview-usage-and-examples/#:~:text=The%20Union%20operator%20combines%20the,has%203%2C4%2C5.>  Note: With Slides-good graphics and example |  |
| 36(P1) | Difference between Delete and Truncate | What DELETE does  What truncate does  When to use delete and truncate with e.g. and demo.  Note: With Slides-good graphics and example | 5 |
| 37(P1) | Queries and Sub-queries | Dis on Queries (SELECT)  Disc on Sub-queries (What is a sub-query)  Why to use sub-queries.  E.g. of Using of sub-query  Note: With Slides-good graphics and example |  |
| 38(P1) | Queries and Nested Queries | Dis on nested Queries (SELECT)  Disc on nested queries (What is a nested query)  Why to use nested queries.  E.g. of Using of nested query  Note: With Slides-good graphics and example |  |
| 39(P1) | Queries and sub-queries using Group by clause | Dis on Group by (SELECT)  Disc on Group by)  Why to use group by clause.  E.g. of Using of group by  Note: With Slides-good graphics and example |  |
| 40(P1) | Having | Dis on Having (SELECT)  Disc on Having )  Why to use having clause.  E.g. of Using of having clause  Note: With Slides-good graphics and example |  |
| 41(P1) | Queries and sub-queries using in and not in clause | Dis on in and not in clause (SELECT)  Disc on in and not in clause)  Why to use in and not in clause.  E.g. of Using of in and not in clause  Note: With Slides-good graphics and example |  |
| 42(P1) | Queries and sub-queries using exists and not exists | Dis on exists and not exists clause (SELECT)  Disc on exists and not exists clause)  Why to use exists and not exists clause.  E.g. of Using of exists and not exists in clause  Note: With Slides-good graphics and example |  |
| 43(P1) | Aggregate Functions | Dis on SUM, AVG, COUNT , MIN, MAX  2-3 e.g. of each  Live DEMO;  Note: With Slides-good graphics and example |  |
| 44(P1) | Introduction to PL/SQL | Dis on PL/SQL  Note: With Slides-good graphics and example |  |
| 45(P1) | Using stored procedure | Need for SP  How to create Stored Procedure  Namening conventions of SP.  IN, OUT, and INOUT parameters  Note: With Slides-good graphics and example |  |
| 46(P1) | Using Functions | Need for Functions  How to create Functions  Namening conventions of Functions.  Note: With Slides-good graphics and example |  |
| 47(P1) | Using Triggers | Need for Triggers  How to create Triggers  Namening conventions of Triggers.  Note: With Slides-good graphics and example |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |