Sr. Software Engineer (SSE)

ABSTRACT

This is one of the subject from my personal notes series named “Coding-With-Arqam” that I am developing from the start of my professional development career.

Subject

Database

DATABASE

--> General Points:

-> We can use distinct in many cases for optimization.
-> First invocation (base case), recursive member (calling function again with condition).
-> An initial query that returns the base result set of the CTE. The initial query is called an anchor member.
-> A recursive query that references the common table expression, therefore, it is called the recursive member.
-> The sequesce of running of sql query (like from, where, select etc.) is for optimization.
-> A CTE cannot be nested while Views can be nested.
-> View once declared can be used for any number of times but CTE cannot be used.
-> Stored procedures have a precompiled execution plan,
where as functions are not. - Because of precompiled plan, for routines, stored procedure is preferred a lot.
A Function can be used in the SQL Queries while a procedure cannot be used in SQL queries.
that cause a major difference b/w function and procedures
-> char VS varchar = every element will take same space as described eg:10. Varchar length varies.
-> Recursive query aborted after 1001 iterations.
-> Image processing: requires optimization to great level beacuse of excess data.
-> InnoDB is a general-purpose storage engine that balances high reliability and high performance.
-> Storage engines (underlying software component) are MySQL components.
-> Aggregate functions(sum, avg, min, max, etc) takes long time.
-> Unique is better than Distinct.
-> Cons:
-> Updates and inserts are more expensive.
-> Denormalization can make update and insert code harder to write.
-> Data may be inconsistent . Which is the “correct” value for a piece of data?
-> Data redundancy necessitates more storage.

-> The time complexity of two dimentional array is O(N), which means its time complexity is linear.
-> Command to show current processes: show processlist;
-> SHOW VARIABLES LIKE "secure_file_priv";
-> We cannot use 'use database xyz' within the stored procedures.
-> SHOW PROCESSLIST;
-> Show triggers;
-> Show events;
-> Show tables;
-> Show databases;
-> describe Section;
-> to how many levels we can use CTE in SQL ? // i think 16.
*-> Use "Explain" keyword before the query to see how that query works. Like "Explain Select * from Emp;". (NOTE: VERY USEEFUL)*

--> Major Topics:
Normalization, SP, CTE, Temp_Tables, Views, Joins, Indexing, Optimizations, limitations.

--> Data Types:
-> Numeric:
-> TINYINT
-> SMALLINT
-> MEDIUMINT
-> INT
-> BIGINT
-> FLOAT(m,d)
-> DOUBLE(m,d)
-> DECIMAL(m,d)
-> BIT(m)
-> BOOL
-> BOOLEAN
-> Date and Time:
-> YEAR[(2|4)]
-> DATE
-> TIME
-> DATETIME
-> TIMESTAMP(m)
-> String:
-> CHAR(size)

-> VARCHAR(size)
-> TINYTEXT(size)
-> TEXT(size)
-> MEDIUMTEXT(size)
-> LONGTEXT(size)
-> BINARY(size)
-> VARBINARY(size)
-> ENUM
-> SET

--> Variables:
-> User-Defined Variable:
-> Sometimes, we want to pass values from one statement to another statement.
-> The user-defined variable enables us to store a value in one statement and later can refer it to another statement.
-> MySQL provides a SET and SELECT statement to declare and initialize a variable.
-> The user-defined variable name starts with @ symbol.
-> The user-defined variables are not case-sensitive such as @name and @NAME; both are the same.
-> We can assign the user-defined variable into limited data types like integer, float, decimal, string, or NULL.
-> Syntax:
-> SET @var_name = value; // To set the value
-> SELECT @name;   // To show the value.
-> Local Variable:
-> It is a type of variable that is not prefixed by @ symbol.
-> The local variable is a strongly typed variable.
-> The scope of the local variable is in a stored program block in which it is declared.
-> MySQL uses the DECLARE keyword to specify the local variable.
-> The DECLARE statement also combines a DEFAULT clause to provide a default value to a variable.
-> If you do not provide the DEFAULT clause, it will give the initial value NULL.
-> It is mainly used in the stored procedure program.
-> Example:
-> DECLARE total_price Oct(8,2) DEFAULT 0.0;
-> DECLARE A INT DEFAULT 100;
-> System Variable:
-> Predefined variables.
-> MySQL server gives a bunch of system variables such as GLOBAL, SESSION, or MIX types.

--> User Management:
-> Create User:
-> The MySQL user is a record in the USER table of the MySQL server that contains the login information, account privileges, and the host information for MySQL account.
-> It is essential to create a user in MySQL for accessing and managing the databases.
-> It provides authentication, SSL/TLS, resource-limit, role, and password management properties for the new accounts.
-> If you want to use the Create User, it is required to have a global privilege of Create User statement or the INSERT privilege for the MySQL system schema.
-> Why Did Users require in MySQL server:
-> When the MySQL server installation completes, it has a ROOT user account only to access and manage the databases.
-> But, sometimes, you want to give the database access to others without granting them full control.
-> Syntax:
-> CREATE USER [IF NOT EXISTS] account_name IDENTIFIED BY 'password';
-> In the above syntax, the account_name has two parts one is the username, and another is the hostname, which is separated by @ symbol. Here, the username is the name of the user, and the hostname is the name of the host from which the user can connect with the database server.
-> username@hostname
-> The hostname is optional Like "username@%"
-> Steps:
-> Open the MySQL server by using the mysql client tool.
-> Enter the password for the account and press Enter.
-> select user from mysql.user; // Will show all the users
-> CREATE USER IF NOT EXISTS adam@localhost IDENTIFIED BY 'jtp123456';
-> Grant Privileges to the users:
-> ALL PRIVILEGES: It permits all privileges to a new user account.
-> CREATE: It enables the user account to create databases and tables.
-> DROP: It enables the user account to drop databases and tables.
-> DELETE: It enables the user account to delete rows from a specific table.
-> INSERT: It enables the user account to insert rows into a specific table.
-> SELECT: It enables the user account to read a database.
-> UPDATE: It enables the user account to update table rows.
-> Example:
*-> GRANT CREATE, SELECT, INSERT ON * . * TO peter@localhost;*

```
                    -> Note: We can also grant privileges in the form of role like dba,admin,etc.
                    -> Note: We can also give privileges like max number of queries that the user can run per hour, etc.
          -> Drop User:
                    -> DROP USER 'account_name';
          -> Steps to Change User Password:
                    -> USE mysql;
                    -> UPDATE user SET password = PASSWORD('jtp12345') WHERE user = 'peter' AND host = 'localhost';
                    -> FLUSH PRIVILEGES;

--> Database:
          -> Create :
                    -> CREATE DATABASE [IF NOT EXISTS] database_name [CHARACTER SET charset_name] [COLLATE
collation_name];
                    -> Explanation:
                              -> database_name :  Name of the database
                              -> charset_name  :  It is optional. It is the name of the character set to store every character in a string.
                              -> collation_name:  It is optional that compares characters in a particular character set.
                    -> Example:
                              -> CREATE DATABASE employeesdb;
          -> SELECT:
                    -> USE database_name;
          -> Show/List:
                    -> SHOW DATABASES;
          -> DROP:
                    -> DROP SCHEMA [IF EXISTS] database_name;
          -> COPY:
                    ->
                    ->
--> Table & Views:
          -> Create:
                    -> Example:
                              CREATE TABLE employee_table(
                                        id int NOT NULL AUTO_INCREMENT,
                                        name varchar(45) NOT NULL,
                                        occupation varchar(35) NOT NULL,
                                        age int NOT NULL,
                                        PRIMARY KEY (id)
                              );
          -> ALTER:
                    -> ADD a column in the table:
                              -> ALTER TABLE cus_tbl ADD cus_age varchar(40) NOT NULL;
                    -> Add multiple columns in the table:
                              -> Example:
                                        ALTER TABLE cus_tbl
                                        ADD cus_address varchar(100) NOT NULL
                                        AFTER cus_surname,
                                        ADD cus_salary int(100) NOT NULL
                                        AFTER cus_age ;
                    -> MODIFY column in the table:
                              -> The MODIFY command is used to change the column definition of the table.
                              -> Syntax:
                                        ALTER TABLE table_name
                                        MODIFY column_name column_definition
                                        [ FIRST | AFTER column_name ];
                              -> Example:
                                        -> ALTER TABLE cus_tbl MODIFY cus_surname varchar(50) NULL;
                    -> DROP column in table:
                              -> ALTER TABLE table_name  DROP COLUMN column_name;
                    -> RENAME column in table:
                              ->  ALTER TABLE  cus_tbl
                                        CHANGE COLUMN cus_surname cus_title
                                        varchar(20) NOT NULL;
                    -> RENAME table:
                              -> ALTER TABLE table_name  RENAME TO new_table_name;
          -> Show/List Tables:
                    -> SHOW TABLES;
                    -> SHOW FULL TABLES;
                    -> SHOW TABLES IN database_name;  OR SHOW TABLES FROM database_name;
                    -> SHOW dbName.TABLES;
                    -> SHOW TABLES LIKE pattern;
                    -> SHOW TABLES IN mysql LIKE "time%";
          -> Rename:
                    -> RENAME old_table TO new_table;
          -> TRUNCATE:
```

-> TRUNCATE TABLE table_name;
-> MYSQL TRUNCATE statement removes the complete data without removing its structure.
-> The TRUNCATE TABLE statement is used when you want to delete the complete data from a table without removing the table structure.
-> DROP :
-> DROP TABLE table_name;
-> Temporary Table:
-> Allows us to keep temporary data.
-> We can reuse this table several times in a particular session.
-> This table is visible and accessible only for the current session.
-> A temporary table can be created by the user with the same name as a normal table in a database.
But then permanenet table will not be accessible till temp table with same name exists.
-> Copy/Clone/Duplicate Table:

```
->  CREATE TABLE IF NOT EXISTS new_table_name
        SELECT column1, column2, column3
        FROM existing_table_name
        WHERE condition;
```

-> Add/Delete Column:

```
-> ALTER TABLE table_name
        ADD COLUMN column_name column_definition [FIRST|AFTER existing_column];
```

-> Views:
-> A view is a database object that has no values.
-> Virtual table created by a query by joining one or more tables.
-> It is operated similarly to the base table but does not contain any data of its own.
-> If we update the main table, view also get update automatically and vice versa.
-> SELECT in views points to the original table and opens the original tbale within the view.
-> UPDATE command is not applicable to all views.
-> An updatable view is one which allows performing a UPDATE command on itself without affecting any other table.
-> Views in SQL are kind of virtual tables.
-> A view also has rows and columns as they are in a real table in the database.
-> Views can hide the complexity of data.
-> Views take very little space to store; the database contains only the definition of a view, not a copy of all the data that it presents..
-> views can provide extra security.
-> Updating Views:
-> There are set of conditions that must be fulfilled to update a view like view should be created using a single table if we want to update and doesnot contains any null value in it.
-> Syntax:
-> CREATE VIEW trainer AS SELECT course_name,trainer FROM courses;
-> Update:
-> ALTER VIEW view_name AS   SELECT columns   FROM table   WHERE conditions;
-> Drop:
-> DROP VIEW [IF EXISTS] view_name;
-> Create View with JOIN Clause:

```
->  CREATE VIEW Trainer
        AS SELECT c.course_name, c.trainer, t.email
        FROM courses c, contact t
        WHERE c.id = t.id;
```

-> When can a view be updated:
-> The view is defined based on one and only one table.
-> The view must include the PRIMARY KEY of the table based upon which the view has been created.
-> The view should not have any field made out of aggregate functions.
-> The view must not have any DISTINCT clause in its definition.
-> The view must not have any GROUP BY or HAVING clause in its definition.
-> The view must not have any SUBQUERIES in its definitions.
-> If the view you want to update is based upon another view, the later should be updatable.
-> Any of the selected output fields (of the view) must not use constants, strings or value expressions.

--> Constraints:
-> UNIQUE: Helps to ensure that all the values in a column are different. (While creation)
-> DISTINCT: Distinct helps to remove all the duplicate records when retrieving the records. (While retriving)
-> NOT NULL
-> CHECK
-> DEFAULT
-> PRIMARY KEY
-> AUTO_INCREMENT
-> INDEX
-> ENUM
-> FOREIGN KEY

--> Indexes:
-> A database index is a data structure that improves the speed of operations in a table.

-> Indexes can be created using one or more columns, providing the basis for both rapid random lookups and efficient ordering of access to records.
-> Lookup table.
-> An index is a data structure that allows us to add indexes in the existing table.
-> It creates an entry for each value of the indexed columns.
-> Primary key is the default index. We called this index as a clustered index.
-> All indexes other than PRIMARY indexes are known as a non-clustered index or secondary index.
-> Generally, we create an index at the time of table creation in the database.
-> Need for Indexing in MySQL:
-> CREATE INDEX [index_name] ON [table_name] (column names)
-> Use "Explain" keyword to see how the query works internally. This will explain index in detail.
-> Drop Index:
-> DROP INDEX index_name ON table_name [algorithm_option | lock_option];
-> Show Indexes:
-> SHOW INDEXES FROM table_name;
-> SHOW INDEXES FROM table_name IN database_name;
-> SHOW INDEXES FROM database_name.table_name;
-> UNIQUE INDEX:
-> MySQL allows another constraint called the UNIQUE INDEX to enforce the uniqueness of values in one or more columns.
-> We can create more than one UNIQUE index in a single table, which is not possible with the primary key constraint.
-> Clauses:
-> https://www.javatpoint.com/mysql-where

--> Functions:
-> substring, distinct , upper, INSTR, LTRIM, RTRIM, length, REPLACE, CONCAT, ilike, BINARY, BETWEEN,

--> Keys:
-> super key (reduced form of composite key + parent key in case of multiple multilevel dependencies)
-> composite key
-> primary key
-> foreign key
-> candidate key

--> Dependencies in db:
-> Transitive-Functional(non key column depends on non-key column),
-> Partial(nonprime attribute is functionally dependent on part of a candidate key. (A nonprime attribute is an attribute that's not part of any candidate key.)
For example, let's start with R{ABCD}, and the functional dependencies AB->CD and A->C), etc.

--> Aggregate Functions:
AVG
COUNT
MAX = if arg is string, it will show the string with max occourences. will show max value in case of integer.
MIN
SUM

--> In which language MySQL has been written?
-> C and C++, and its SQL parser is written in yacc.

--> SQL vs MySQL:
-> SQL is a computer language, whereas MySQL is a software or an application.
-> SQL is used for the creation of database management systems whereas MySQL is used to enable data handling, storing, deleting and modifying data.

--> Tables present in MySQL:
-> MyISAM (Default), Heap, Merge, INNO DB, ISAM

--> MySQL Intallation:
->https://www.javatpoint.com/how-to-install-mysql

--> How to change the MySQL password:
-> ALTER USER 'root'@'localhost' IDENTIFIED BY 'NewPassword';

--> Database Management System:
-> MySQL, Oracle, etc.

--> Types of databases:
-> Centralized db:
-> Central Library of a college.
-> Distributed db:
-> Apache Cassandra, HBase, Ignite, etc.
-> NoSQL db:

-> Mongo db.
-> Cloud db:
-> Amazon Web Services(AWS), Microsoft Azure, Kamatera, PhonixNAP, ScienceSoft, Google Cloud SQL, etc.
-> Relational db:
-> MySQL, Microsoft SQL Server, Oracle, etc.
-> Network db:
->
-> Object Oriented db:
->
-> Heirarchical db:
->
->Personal Database:
->
-> Operational Database:
-> An organization uses operational databases for managing per day transactions.
-> Enterprise Database:
->

--> RDBMS:
-> SQL, MS SQL Server, IBM DB2, ORACLE, My-SQL and Microsoft Access are based on RDBMS.
->Integrity Constraints:
-> Domain constraint, Entity Integrity constraint, Referrential Ingtegrity constraint, Key constraint.

--> DBMS vs RDBMS:
-> Store data as file => Store data in a tabular form
-> Normalization is not present in DBMS => Present
-> Does not apply any security => Apply
-> Types of relational operations:
-> Select, Project, Union, Set Intersection, Set Difference, Catesian Product, Rename.

--> SQL:
-> It is used for storing and managing data in relational database management system (RDMS).
-> Standard language for Relational Database System.
-> All the RDBMS like MySQL, Informix, Oracle, MS Access and SQL Server use SQL as their standard database language.
-> Commands:
-> DDL:
-> Create, Drop, Alter, Truncate
-> DML:
-> Insert, Update, Delete
-> TCL (Transaction Control Language):
-> Commit, RollBack, SavePoint
-> DQL:
-> Select
-> Operators:
-> Arithmetic
-> Comparison
-> Logical
--> Normalization:
-> 1NF:
-> A relation is in 1NF if it contains an atomic value.
-> It converts multiple values to atomic values against each attributes. Like "Ali" hasMany "ContactNo"
-> 2NF:
-> A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
-> Example:
-> Before 2NF:
-> Teacher Table:
-> teacher_id, subject, age
-> After 2NF:
-> Teacher_Details table:
-> teacher_id, age
-> Teacher_Subjects table:
-> teacher_id, subject
-> 3NF:
-> A relation will be in 3NF if it is in 2NF and no transition dependency exists.
-> Boyce Codd normal form (BCNF):
->
-> 4NF:
-> A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
-> 5NF:
-> A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.

--> Order of execution of Queries (Hint: FOOW-GHS-DOT-L):
FROM clause

ON clause
OUTER clause
WHERE clause
GROUP BY clause
HAVING clause
SELECT clause
DISTINCT clause
ORDER BY clause
TOP clause/ LIMIT

--> SQL Commands Category:
DDL = create, alter, drop, rename, truncate, comment.
DML = select, insert, update, delete, merge, call, explain plan, lock table.
DCL = grant, revoke.
TCL = commit, rollback, savepoint, set transaction.

--> Lock and Unlock Tables:
-> MySQL enables client sessions to acquire table locks explicitly for the purpose of cooperating with other sessions for access to tables, or to prevent other sessions from modifying tables during periods when a session requires exclusive access to them.
-> Types: shared(S) locks and exclusive (X) locks.
-> Levels: Row level locking, Table level locking.
-> This is also a time optimization technique.
-> If we lock a table emp with read, you cannot read other tables then.
-> First we lock a table, do transactions on it then unlock it.
-> This may create a deadlocks.
-> This locking level makes these storage engines more suitable for read-only, read-mostly, or single-user applications.

--> Storage Engines:
-> Storage engines (underlying software component) are MySQL components, that can handle the SQL operations for different table types to store and manage information in a database.
-> InnoDB is mostly used general-purpose storage engine and as of MySQL 5.5 and later.
-> Inno DB suppports:
ACID compliant, FOREIGN KEY referential-integrity constraints,
commit, rollback, and crash-recovery capabilities to protect data, row-level locking

--> Clauses:
-> Having:
-> The HAVING clause was initially added to SQL because the WHERE keyword could not be used with aggregate functions.
-> HAVING is typically used with the GROUP BY clause to restrict the groups of returned rows to only those that meet certain conditions.
-> if you use this clause in your query, the index is not used, which can result in a query that doesn't really perform all that well.
-> handle having clause by using where clause to optimize time.
--> Complexity types: (given in descending order)
-> O(n!)
-> O(2^n)
-> O(n^2)
-> O(n log n)
-> O(n)
-> O(log n),O(1)

--> Indexing VS Heap table:
-> Indexing:
-> Def:
Indexing is a way to optimize the performance of a database by minimizing the number of disk accesses required when a query is processed.
-> Strategy:
-> Search key:
Contains a copy of the primary key or candidate key of the table.
These values are stored in sorted order so that the corresponding data can be accessed quickly.
The data may or may not be stored in sorted order.
-> Data Reference or Pointer:
Contains a set of pointers holding the address of the disk block where that particular key value can be found.

-> Types:
-> Clusted:
-> When more than two records are stored in the same file these types of storing known as cluster indexing.
-> Pointed to the location that contains the actual data.
-> Leaf nodes contain the actual data sorted by id.
-> Leaf nodes are doubly linked list.

```
                    -> root contains only one page(array of id of all the records of a table)
                    -> One table can only contains One cluster index.
                    -> It is like PK.
                    -> E.g: Index page of a book.
               -> Non-Clustered or Secondary Indexing:
                    -> Pointed to the location that contains the reference of the actual data.
                    -> Having no PK.
                    -> create index on some field like name.
                    -> Leaf nodes doesnot contain the actual data sorted by id.It contains the row locator(reference).
                    -> E.g: Glossary page of a book(Which pages contains the specific word).

               -> Multilevel Indexing:
     -> Heap Table:
          -> Tables without indexing.

--> Events:
     -> Tasks that run according to a schedule.
     -> Also called scheduled events.
     -> They run in the background unlike to trigers that only runs on insert, update and delete of specific table only.
     -> Beginning and ending at a specific date and time.
     -> show processlist; // command to check event is on or off.
     -> set GLOBAL
     -> Types: after/before insert, before/after update, before/after delete.
     -> E.g: // Runs after every 1 minute
          use `e-commerce`;
          CREATE EVENT IF NOT EXISTS expire_products
          ON SCHEDULE
          EVERY 1 MINUTE
          STARTS CURRENT_TIMESTAMP
          DO
           INSERT INTO productcategory (name) values ('ali');
     -> E.g: // Runs daily from given time.
          CREATE EVENT my_event
           ON SCHEDULE
               EVERY 1 DAY
               STARTS '2014-04-30 00:20:00' ON COMPLETION PRESERVE ENABLE
           DO
               # My query
     -> E.g: // Runs on given date once.
          CREATE EVENT my_event
           ON SCHEDULE
               AT ('2014-04-30 00:20:00'+ INTERVAL 1 DAY) ON COMPLETION PRESERVE ENABLE
           DO
               # My query
     -> ALTER EVENT expire_products DISABLE; // to disable the scheduler event.
     -> show events;
     -> DROP EVENT IF EXISTS expire_products;
     -> Example: // Calling SP within a Event Scheduler;
          DROP EVENT IF EXISTS expire_products;
          CREATE EVENT IF NOT EXISTS expire_products
          ON SCHEDULE
          EVERY 3 SECOND
          STARTS CURRENT_TIMESTAMP
          Do call ard_db_dev.testing();


--> Triggers:
     -> A MySQL event is a task that rums based on a predefined schedule therefore
          sometimes it is referred to as a scheduled event.
          In other words, we can say that MySQL event schedule is a process that runs in the background
          and constantly looks for the events to execute. It is referred to as a temporal trigger because
          they triggered by time and not like triggers that execute based on the table update.
     -> show triggers;
     -> Example:
          -> After Insert, etc.

--> mysqldump:
     ->
     ->

--> Creating table similar to another with data:
     -> Method 1:
          create table new_table
          as
```

```
                    select * from e_commerce_backup.colors;

          -> Method 2:
                    DROP TABLE e_commerce_backup.colors;
                    CREATE TABLE e_commerce_backup.colors  LIKE `e-commerce`.colors;
                    INSERT INTO e_commerce_backup.colors (SELECT * FROM `e-commerce`.colors);
--> Getting all tables in a database:
          ->
                    Use e_commerce_backup;
                    select * from information_schema.tables where TABLE_SCHEMA = 'ard_db_dev';

          -> SHOW TABLES from e_commerce_backup;

--> Prepared Statement:
          ->
          -> Example:
                    DECLARE Query nvarchar(200);
                    SET @Query= 'SELECT * FROM ard_db_dev.ACL';
                    PREPARE stmt3 FROM @Query;
                    EXECUTE stmt3;
                    DEALLOCATE PREPARE stmt3;

          -> Example: PS within SP.
                    CREATE DEFINER=`ard_db_root`@`%` PROCEDURE `ard_db_dev_bk_sp`()
                    BEGIN
                    -- Variables Declarations...!

                    DECLARE tableName CHAR(255);
                    DECLARE myQuery VARCHAR(255) default "";
                    DECLARE counter INT DEFAULT 0;
                    DECLARE check_done INT DEFAULT 0;

                    -- Cursors Declations...!
                    DECLARE tablesList CURSOR FOR select TABLE_NAME from information_schema.tables where TABLE_SCHEMA =
'ard_db_dev';

                    DECLARE CONTINUE HANDLER FOR NOT FOUND SET check_done = 1;

                    DROP DATABASE IF EXISTS ard_db_dev_bk;
                    CREATE DATABASE ard_db_dev_bk;

                    OPEN tablesList;

                     insert_loop: LOOP
                              FETCH tablesList INTO tableName;
                              IF tableName is null Then
                              leave insert_loop;
                              END If;
                              SET @myQuery = CONCAT('create table ard_db_dev_bk.' , tableName , ' as' , ' select * from ard_db_dev.'
, tableName);
                              PREPARE ps_query FROM @myQuery;
                              EXECUTE ps_query;
                              DEALLOCATE PREPARE ps_query;
                              SET counter =  counter + 1;
                              END LOOP insert_loop;
                    CLOSE tablesList;
                    END
```

--> Schema Migration:
-> schema migration refers to the management of incremental,
reversible changes and version control to relational database schemas.
-> A schema migration is performed on a database,
whenever it is necessary to update or revert that database's schema to some newer or older version.

--> Virtual Field:
-> It is in mongo db.
-> a field which is in front-end but not in db.
-> Like we have firstName and lastName in db but we have fullName field too in front-end.

--> Junction table:
-> Bridge Table
-> Join Table
-> Map Table
-> Link Table

-> Cross-Reference Table

MySQL Basic Queries

--> Queries:

1.
```
use programming_shell;
```
2.
```
CREATE DATABASE IF NOT EXISTS developers_career;
```
3.
```
create table category(
id int not null auto_increment primary key,
name varchar(255)
);
```
4.
```
create table answer(
id int not null auto_increment,
answer varchar(255),
question_id int,
PRIMARY KEY (id),
FOREIGN KEY (question_id) REFERENCES question(id)
);
```
5.
```
alter table programming_shell.image add answer_id int;
```
6.
```
ALTER TABLE programming_shell.image ADD FOREIGN KEY (answer_id) REFERENCES answer(id);
```
7.
```
Alter table topicmetadata
add  tag_id int,
add Foreign key (tag_id) References tag(id)
```
8.
```
alter table tag
drop topic_id
```
9.
To test API:
-> use tools like postman, insomnia etc.
-> use cmd command like:
```
    -> curl -X  GET "http://localhost:3000/question/get-all-questions"
    -> curl -X  GET "http://localhost:3000/question/get-question-by-id1" // this is to get by id where id=1
```
10.
```
delete from angular_node_modules_db.endpoint where id > 0 and name like '%sss%';
```

MySQL Advanced Queries

--> Queries:

1.
```
SELECT ardProRoleId,name, AVG(ardProRoleId)
FROM ard_db_stage.ArdProUser
GROUP BY name -- this is compulsory otherwise it will give average of first row
order by id asc;
```

2.
```
SELECT ardProRoleId, count(ardProRoleId)
FROM ard_db_stage.ArdProUser
group by name -- this will show records having same name.
having count(ardProRoleId) > 1;
```

3.
```
ALTER TABLE `document` MODIFY `document_id` INT AUTO_INCREMENT PRIMARY KEY;
```

4.
// CTE are substitute of views.
// There are some situations where we cannot create views due to permission, we can use CTE there.
```
WITH Employee_CTE (EmployeeNumber, Title)
AS // (Query Definition) Don't use ORDER BY, unless you also use as TOP clause, INTO, OPTION clause.
(SELECT NationalIDNumber,
                    JobTitle
 FROM   HumanResources.Employee)
SELECT EmployeeNumber,
            Title
    FROM   Employee_CTE
```

5.
// Temporary tables in SQL.

```
    CREATE TEMPORARY TABLE if not exists top10customers
    SELECT id
    FROM ard_db_stage.ArdProUserDevices
    LIMIT 2;
```

// if workbench(editor) closes, it also get remove.
// query to drop temp table: DROP TEMPORARY TABLE table_name;
// Adv: you can see only, fast (SQL Server has less logging and locking overheads for temporary tables),
Simplicity of coding, Not as fast as table variables, Can not update in functions
// These are stores in TempDb

6.
// Generic recursive CTE.

```
WITH   cte
AS     (SELECT 1 AS n -- anchor member
            UNION ALL
            SELECT n + 1 -- recursive member
            FROM   cte
            WHERE  n < 50 -- terminator
       )
SELECT n
FROM   cte;
```

7.

```
update ard_db_dev.Class set created = '2019-06-11 10:45:43' where id > 0;
```

Note:
below query will give error of using save mode because the column to update and the column in the where clause are same.

```
update ard_db_stage.UserReferralRequest set ardProUserId = '99' where ardProUserId = 3;
```

8.
// SP to copy data from one table to another of two different databses.

```
    CREATE DEFINER=`ard_db_root`@`%` PROCEDURE `Stage_Dev_.GoalObjective`()
    BEGIN
    DELETE FROM ard_db_dev.GoalObjective WHERE id>0;

    alter table ard_db_dev.GoalObjective AUTO_INCREMENT = 1;

    INSERT INTO ard_db_dev.GoalObjective
    (id, objectiveNo, description, goalId)
    SELECT id, objectiveNo, description, goalId
    FROM ard_db_stage.GoalObjective;
    END
```

9.
// Reccursive CTE in SP to find leaf nodes of N-Level categories.

```
    CREATE DEFINER=`cp_sc`@`%` PROCEDURE `getLeafCategories`(IN categoryId INT)
    BEGIN
    WITH recursive Employee_CTE(id, productCategoryId) AS
    (
      SELECT id, productCategoryId from productCategory where id = categoryId
      UNION ALL
      SELECT e.id, e.productCategoryId
      from productCategory e
      INNER JOIN Employee_CTE c ON e.productCategoryId = c.id
    )
    select id AS leafCategoryId from Employee_CTE where id not in (select productCategoryId from productCategory);
    END
```

10.
// Scheduler Event:

```
    use `e-commerce`;
    CREATE EVENT IF NOT EXISTS expire_products
    ON SCHEDULE
    EVERY 1 MINUTE
    STARTS CURRENT_TIMESTAMP
    DO
      INSERT INTO productcategory (name) values ('ali');
```

11.
// Calling SP within a Event Scheduler;

```
DROP EVENT IF EXISTS expire_products;
CREATE EVENT IF NOT EXISTS expire_products
ON SCHEDULE
EVERY 3 SECOND
STARTS CURRENT_TIMESTAMP
Do call ard_db_dev.testing();
```

Hackerrank Queries

-- Observe the output of all the below queries.

- *select distinct name from students where marks > 75 order by right(name,3);*
- *select distinct name from `e-commerce`.productcategory where productCategoryId > 10 group by right(Name,3) order by right(Name,3) asc;*
- *SELECT IF(g.Grade<8, NULL, s.Name), g.Grade, s.Marks FROM Students AS s JOIN Grades AS g ON s.Marks BETWEEN g.Min_Mark AND g.Max_Mark ORDER BY g.Grade DESC, s.Name, s.Marks;*
- *select sum(population) from CITY where COUNTRYCODE like "JPN"*
- *SELECT NAME FROM STUDENTS WHERE MARKS > 75 ORDER BY RIGHT(NAME, 3), ID ASC;*
- *SELECT IF(g.Grade<8, NULL, s.Name), g.Grade, s.Marks FROM Students AS s JOIN Grades AS g ON s.Marks BETWEEN g.Min_Mark AND g.Max_Mark ORDER BY g.Grade DESC, s.Name, s.Marks;*
- *SELECT DISTINCT city FROM STATION WHERE right(city,1) not in ("a","e","i","o","u", "A","E","I","O","U") or left(city,1) not in ("a","e","i","o","u","A","E","I","O","U");*
- *SELECT IF(A+B>C AND A+C>B AND B+C>A, IF(A=B AND B=C, 'Equilateral', IF(A=B OR B=C OR A=C, 'Isosceles', 'Scalene')), 'Not A Triangle') FROM TRIANGLES;*
- *SELECT IF(Marks + 20 > 100, "Msg to display", name) FROM hackerrrank.students;*
- *SELECT CEIL(AVG(Salary) - AVG(REPLACE(Salary, '0', ''))) FROM EMPLOYEES;*
- *SELECT REPLACE(Marks, 8,'') FROM hackerrrank.students;*
- *SELECT name AS 'Label', JSON_OBJECTAGG(Name, productcategoryId) AS 'JSON' FROM `e-commerce`.productcategory group by label;*
- *select (count(city)- count(distinct city)) from Station;*
- *select name from `e-commerce`.productcategory having max(length(name));*
- *select name from `e-commerce`.productcategory having max(length(name)) order by name asc limit 1;*
- *select name,LENGTH(name) from `e-commerce`.productcategory order by Length(name) asc, name limit 1; // same as above but optimized*
- *select sum(population) from city where district like '%California%';*
- *SELECT * FROM ard_db_stage.Goal where id between 299 and 306;*
- *SELECT name, SUM(productCategoryId) FROM `e-commerce`.productcategory GROUP BY (name);*
- *SELECT name, SUM(productCategoryId) FROM `e-commerce`.productcategory GROUP BY (productCategoryId);*

Reference Links

- *https://www.techbeamers.com/sql-query-questions-answers-for-practice/*
- *http://www.mysqltutorial.org/mysql-temporary-table/*
- *https://www.designcise.com/web/tutorial/what-is-the-order-of-execution-of-an-sql-query*
- *https://www.periscopedata.com/blog/sql-query-order-of-operations*
- *http://www.mysqltutorial.org/mysql-table-locking/*
- *https://www.geeksforgeeks.org/sql-views/*

- *http://www.sql-join.com/sql-join-types*
- *https://www.datacamp.com/community/tutorials/sql-tutorial-query*
- *https://www.sqlshack.com/what-is-the-difference-between-clustered-and-non-clustered-indexes-in-sql-server/*
- *https://aboutsqlserver.com/2010/09/22/indexes-structure/*
- *https://www.geeksforgeeks.org/sql-queries-on-clustered-and-non-clustered-indexes/*
- *https://dba.stackexchange.com/questions/64208/scheduling-an-event-every-day-at-a-given-time*
- *https://www.liquidweb.com/kb/how-to-back-up-mysql-databases-from-the-command-line/*
- *http://www.niteshluharuka.com/how-to-execute-a-query-stored-in-a-variable-in-stored-procedure/*
- *http://g2pc1.bu.edu/~qzpeng/manual/MySQL%20Commands.htm*
- *https://www.javatpoint.com/mysql-tutorial (MySQL on JavaTpoint)*