**MY SQL ASSIGNMENT(DDL &DML COMMANDS) ANSWERS**

1. Create database if not exists assignment;
2. I directly run this assignment\_tables.sql in MySQL work bench..
3. Create table authors (authorid integer, name varchar (255));
4. insert into authors (authorid, name) values (1,"J K Rowling"), (2,"Thomas Hardy"), (3,"Oscar Wilde"), (4,"Sidney Sheldon"), (5,"Alistair Maclean"), (6,"Jane Austen");
5. insert into authors (authorid, name) values (7,"Harvey Deitel"), (8,"Paul Deitel"), (9," Andrew Goldberg"), (10," David Choffnes");
6. update authors set name="Alistair McNeal" where name="Alistair Maclean";
7. create table Books (bookid integer, title varchar (255), authorid integer);
8. insert into Books (bookid, title, authorid)

-> values (1,"Harry Potter and the Philosopher's stone",1),

-> (2,"Harry Potter and the Chamber of Secrets",1),

-> (3,"Harry Potter and the Half-Blood Prince",1),

-> (4,"Harry Potter and the Goblet of Fire",1),

-> (5,"Night Without End",5),

-> (6,"Fear is the Key",5),

-> (7,"Where Eagles Dare",5),

-> (8,"Sense and Sensibility",6),

-> (9,"Pride and Prejudice",6),

-> (10,"Emma",6), (11,"Random Book",22);

b. delete from Books where title="Random Book";

5. a) rename table Books to Favbooks;

b) rename table authors to Favauthors;

6. b) create table Suppliers (supplier\_id integer primary key Auto\_increment, supplier\_name varchar (255), location varchar (255));

a) create table Product (product\_id integer primary key auto\_increment, product\_name varchar (255) NOT NULL UNIQUE, description varchar (255), supplier\_id references Suppliers(supplier\_id));

c) create table Stocks (id integer primary key auto\_increment, product\_id integer references product(product\_id), balance\_stock integer);

7**. inserting into Suppliers:**

insert into Suppliers (supplier\_id, supplier\_name, location) values(1,"aaa","hyd"), (2,"bbb","hyderabad"), (3,"ccc","banglore"), (4,"ddd","delhi"), (5,"abc","delhi");

**inserting into Product:**

insert into Product (product\_id, product\_name, description, supplier\_id) values (1,"Microsoft Lumia","mabe by microsoft",1), (2,"HTC One","made by HTC",2), (3,"Nexus","made by nexus",1), (4,"iphone","made by apple",5), (5,"lenovo Thinkpad","made by lenovo",6);

**inserting into Stock:**

insert into stock (id, product\_id, balance\_stock) values (100,1,1000), (null,4,2000);

insert into stock (product\_id, balance\_stock) values (5,1000), (4,2000), (2,3000);

8. alter table Suppliers modify supplier\_name varchar (255) NOT NULL UNIQUE;

9. a) alter table emp add deptno integer;

b) update table emp set deptno=20 where emp\_no %2=0;

update table emp set deptno=30 where emp\_no %3=0;

update table emp set deptno=40 where emp\_no%4=0;

update table emp set deptno=50 where emp\_no%5=0;

update table emp set deptno=10 where emp\_no %2=0 || emp\_no %3=0|| emp\_no%4=0|| emp\_no%5=0;

10. create UNIQUE INDEX HINDEX

-> USING HASH

-> ON emp(emp\_no);

**Assignment\_part2 DML**

1. select \*from employee where deptno in (10) and salary>3000;

+-------+--------+---------+--------+----------+

| empid | fname | lname | deptno | salary |

+-------+--------+---------+--------+----------+

| 800 | George | Clooney | 10 | 10000.00 |

+-------+--------+---------+--------+----------+

1. alter table students add Grade varchar (255);

🡪update table students set Grade="Distinction" where marks between 80 and 100;

🡪update students set Grade="First Class" where marks between 50 and 80;

🡪 update students set grade="Second Class" where marks between 40 and 50;

a) select count (\*) from students where grade="First Class";

+-------------+

| count (\*) |

+-------------+

| 13 |

+--------------+

b) select count (\*) from students where grade="Distinction";

+------------+

| count (\*) |

+-------------+

| 10 |

+-------------+

1. select distinct id, city from station where id%2=0;
2. select count(city) from station;

+-------------+

| count(city) |

+-------------+

| 501 |

+-------------+

select count (distinct CITY) from STATION;

+----------------------+

| count (distinct CITY) |

+----------------------+

| 471 |

+----------------------+

select (count (CITY)- count (distinct CITY)) from STATION;

+------------------------------------------------+

| (count (CITY)- count (distinct CITY)) |

+------------------------------------------------+

| 30 |

+-------------------------------------------------+

1. a) SELECT DISTINCT (city) FROM station

-> WHERE city LIKE 'a%'

-> UNION

-> SELECT DISTINCT (city) FROM station

-> WHERE city LIKE 'e%'

-> UNION

-> SELECT DISTINCT (city) FROM station

-> WHERE city LIKE 'i%'

-> UNION

-> SELECT DISTINCT (city) FROM station

-> WHERE city LIKE 'o%'

-> UNION

-> SELECT DISTINCT (city) FROM station

-> WHERE city LIKE 'u%';

(OR)

SELECT DISTINCT CITY

-> FROM STATION

-> WHERE LEFT(CITY,1) IN ('a', 'e', 'i', 'o', 'u');

b) SELECT DISTINCT CITY

-> FROM STATION

-> WHERE LEFT(CITY,1) IN ('a', 'e', 'i', 'o', 'u') and RIGHT(CITY,1) IN ('a', 'e', 'i', 'o', 'u');

+--------------------------+

| CITY |

+---------------------------+

| Upperco |

| Aguanga |

| Amo |

| Eleele |

| Oconee |

| Amazonia |

| Aliso |

| Andersonville |

| Arkadelphia |

| Eriline |

| Eastlake |

| Arispe |

| Ermine |

| Eufaula |

| Osborne |

| Oshtemo |

| Archie |

| Alpine |

| Ojai |

| Orange |

| Urbana |

| Alba |

| Eureka |

| Eskridge |

| Ozona |

| Acme |

+--------------------------+

c) SELECT DISTINCT CITY

-> FROM STATION

-> WHERE LEFT(CITY,1) NOT IN ('a', 'e', 'i', 'o', 'u');

d) SELECT DISTINCT CITY

-> FROM STATION

-> WHERE LEFT(CITY,1) NOT IN ('a', 'e', 'i', 'o', 'u') and RIGHT(CITY,1) NOT IN ('a', 'e', 'i', 'o’, 'u');

7) SELECT CONCAT (FIRST\_NAME,’ ‘, LAST\_NAME) AS NAME, HIRE\_DATE, SALARY FROM EMP WHERE SALARY>2000 AND HIRE\_DATE>20200201;

8) SELECT DEPTNO, SUM(SALARY) FROM EMP GROUP BY DEPTNO ORDER BY DEPTNO;

+-------------+--------------------+

| DEPTNO | SUM(SALARY)|

+-------------+--------------------+

| 10 | 219800.00 |

| 20 | 94200.00 |

| 30 | 75150.00 |

| 40 | 79199.00 |

| 50 | 46957.50 |

+-------------+--------------------+

9) SELECT COUNT (\*) FROM CITY WHERE POPULATION>100000;

+------------------+

| COUNT (\*) |

+-------------------+

| 11 |

+-------------------+

10) SELECT SUM(POPULATION) FROM CITY WHERE DISTRICT="CALIFORNIA";

+---------------------------+

| SUM(POPULATION) |

+---------------------------+

| 339002 |

+---------------------------+

11) SELECT COUNTRYCODE, AVG(POPULATION) FROM CITY GROUP BY COUNTRYCODE ORDER BY COUNTRYCODE;

+----------------------+---------------------------+

| COUNTRYCODE | AVG(POPULATION) |

+----------------------+---------------------------+

| JPN | 175839.2000 |

| NLD | 593321.0000 |

| USA | 120225.8750 |

+-----------------------+--------------------------+

**ASSIGNMENT\_PART3\_ROUTINES**

1. CREATE DEFINER=`root`@`localhost` PROCEDURE `ORDER\_PROCEDURE` (MONTH VARCHAR (20), YEAR INTEGER)

BEGIN

SELECT ORDERNUMBER, ORDERDATE, STATUS

FROM ORDERS

WHERE substring (MONTHNAME(ORDERDATE),1,3) =MONTH AND YEAR(ORDERDATE)=YEAR;

END

🡪 CALL ORDER\_PROCEDURE("AUG",2004);

+-----------------------+------------------+------------+

| ORDERNUMBER | ORDERDATE | STATUS |

+-----------------------+------------------+-------------+

| 10276 | 2004-08-02 | Shipped |

| 10277 | 2004-08-04| Shipped |

| 10278 | 2004-08-06 | Shipped |

| 10279 | 2004-08-09 | Shipped |

| 10280 | 2004-08-17 | Shipped |

| 10281 | 2004-08-19 | Shipped |

| 10282 | 2004-08-20 | Shipped |

| 10283 | 2004-08-20 | Shipped |

| 10284 | 2004-08-21 | Shipped |

| 10285 | 2004-08-27 | Shipped |

| 10286 | 2004-08-28 | Shipped |

| 10287 | 2004-08-30 | Shipped |

+----------------------+---------------------+-------------+

1. **A) CANCELLATIONS\_PROCEDURE() CREATION:**

CREATE DEFINER=`root`@`localhost` PROCEDURE `CANCELLATIONS\_procedure` ()

BEGIN

CREATE TABLE CANCELLATIONS

(

ID INTEGER primary KEY AUTO\_INCREMENT,

CUSTOMERNUMBER INTEGER references ORDERS(CUSTOMERNUMBER),

ORDERNUMBER INTEGER references ORDERS(ORDERNUMBER),

COMMENTS VARCHAR (255) references ORDERS(COMMENTS),

STATUS VARCHAR (255) references ORDERS(STATUS)

);

END

---> CALL **CANCELLATIONS\_PROCEDURE ();**

**B) CANCELLATION\_CURSOR () CREATION:**

CREATE DEFINER=`root`@`localhost` PROCEDURE `CANCELLATION\_CURSOR` ()

BEGIN

DECLARE CNUMBER INTEGER DEFAULT 0;

DECLARE ONUMBER, NF\_RAISED INTEGER DEFAULT 0;

DECLARE OCOMMENT VARCHAR (255);

DECLARE OSTATUS VARCHAR (255);

DECLARE CANCEL\_CURSOR CURSOR FOR SELECT CUSTOMERNUMBER, ORDERNUMBER, COMMENTS, STATUS FROM ORDERS WHERE STATUS="CANCELLED";

declare continue handler for NOT FOUND

begin

set nf\_raised=1;

end;

OPEN CANCEL\_CURSOR;

MYLOOP: LOOP

FETCH CANCEL\_CURSOR INTO CNUMBER, ONUMBER, OCOMMENT, OSTATUS;

if nf\_raised=1 then

leave MYloop;

end if;

IF OSTATUS='CANCELLED'THEN

INSERT INTO CANCELLATIONS

(CUSTOMERNUMBER, ORDERNUMBER, COMMENTS, STATUS)

VALUES (CNUMBER, ONUMBER, OCOMMENT, OSTATUS);

END IF;

END LOOP MYLOOP;

CLOSE CANCEL\_CURSOR;

END

🡪 **CALL CANCELLATION\_CURSOR ();**

1. **A)**

Create Definer=`Root’@’localhost` Function `PURCHASE\_function` (Cno Int)

Returns Varchar (50) Charset Utf8mb4

Begin

Declare Totalamt Integer;

Declare Purchase\_status Varchar (50);

Select Sum (Amount) Into Totalamt From Payments

Group By Customernumber Having Customernumber=Cno;

Select Status as purchase\_status into Purchase\_status From Customers Where Customernumber=Cno;

If Totalamt<25000 Then

Set Purchase\_status="Silver";

Elseif Totalamt Between 25000 And 50000 Then

Set Purchase\_status="Gold";

Elseif Totalamt>50000 Then

Set Purchase\_status="Platinum";

End If;

Return Purchase\_status;

End

🡪 select assignment. PURCHASE\_function (144) as purchase\_status;

+-------------------------------+

| purchase\_status |

+--------------------------------+

| GOLD |

+---------------------------------+

b)

select customernumber as customerid, customername, assignment. PURCHASE\_function(customernumber) as purchase\_status from customers;

+------------+--------------------------------+----------+

| customerid | customername | status |

+------------+--------------------------------+----------+

| 103 | Atelier graphique | SILVER |

| 112 | Signal Gift Stores | PLATINUM |

| 114 | Australian Collectors, Co. | PLATINUM |

| 119 | La Rochelle Gifts | PLATINUM |

| 121 | Baane Mini Imports | PLATINUM |

| 124 | Mini Gifts Distributors Ltd. | PLATINUM |

| 125 | Havel & Zbyszek Co | NULL |

| 128 | Blauer See Auto, Co. | PLATINUM |

| 129 | Mini Wheels Co. | PLATINUM |

| 131 | Land of Toys Inc. | PLATINUM |

| 141 | Euro+ Shopping Channel | PLATINUM |

.

.

.

4. )BEGIN

declare L\_CID, L\_credit, NF\_RAISED integer;

declare l\_status varchar (50);

declare c1 cursor for select CUSTOMERNUMBER, creditlimit, status from customers;

declare continue handler for NOT FOUND

BEGIN

SET NF\_RAISED=1;

END;

DECLARE CONTINUE HANDLER FOR sqlexception

BEGIN

get diagnostics condition 1 @message\_text=message\_text;

if L\_STATUS="SILVER" THEN

UPDATE CUSTOMERS SET CREDITLIMIT=60000 WHERE CUSTOMERNUMBER=@MESSAGE\_TEXT;

ELSEif L\_STATUS="PLATINUM" THEN

UPDATE CUSTOMERS SET CREDITLIMIT=100000 WHERE CUSTOMERNUMBER=@MESSAGE\_TEXT;

END IF;

END;

open c1;

myloop: loop

fetch c1 into l\_cid, l\_credit, l\_status;

IF NF\_RAISED=1 THEN

LEAVE MYLOOP;

END IF;

if ((L\_credit>60000) and (l\_status="silver”)) then

signal sqlstate '45000'

SET MESSAGE\_TEXT=L\_CID;

ELSEif ((l\_credit<100000) and (l\_status="PLATINUM”)) then

signal sqlstate '45000'

SET MESSAGE\_TEXT=L\_CID;

END IF;

END LOOP MYLOOP;

CLOSE C1;

END

🡪 call assignment. credit\_procedure ();

Another solution for 4th question:

4)

BEGIN

Declare nf\_raised integer default 0;

Declare lcust\_number integer;

Declare lcust\_name varchar (255);

Declare l\_creditlimit integer;

Declare Lpurchase\_status varchar (255);

Declare cred\_cur cursor for select customernumber, customername, creditlimit, PURCHASE\_function(customernumber) as purchase\_status From customers;

Declare continue handler for not found

Begin

Set nf\_raised=1;

End;

Declare continue handler for sqlstate ‘45000’

Begin

Set SQL\_SAFE\_UPDATES=0;

UPDATE customers set creditlimit=1000000 where customernumber= lcust\_number;

End;

Declare continue handler for sqlstate ‘45001’

Begin

Set SQL\_SAFE\_UPDATES=0;

UPDATE customers set creditlimit=60000 where customernumber= lcust\_number;

End;

Open cred\_cur;

Credloop: loop

Fetch cred\_cur into lcust\_number, lcust\_name, l\_creditlimit, Lpurchase\_status;

If nf\_raised=1 then

Leave Credloop;

End if;

If Lpurchase\_status=’platinum’ and l\_creditlimit<1000000 then

Signal sqlstate ‘45000’;

Elseif Lpurchase\_status=’silver’ and l\_creditlimit>60000 then

Signal sqlstate ‘45001’;

End if;

End loop Credloop;

Close cred\_cur;

End

5) **on delete cascade**

CREATE DEFINER=’ROOT’@’LOCALHOST’ TRIGGER

‘Movies\_BEFORE\_DELETE’ BEFORE DELETE ON ‘movies’ FOR EACH ROW

Begin

Delete from rentals where movieid=old.id;

End

**On update cascade**

CREATE DEFINER=’ROOT’@’LOCALHOST’ TRIGGER

‘Movies\_BEFORE\_UPDATE’ BEFORE UPDATE ON ‘movies’ FOR EACH ROW

Begin

UPDATE rentals set movieid=new.id where movieid=old.id;

END