**Assignment 1**

1. create a database called 'assignment' (Note please do the assignment tasks in this database)

CREATE DATABASE assignment;

2. Create the tables from assignment\_tables.sql and enter the records as specified in it.

use assignment;

drop table if exists movies;

create table movies (id integer, title varchar(50), category varchar(25));

drop table if exists customers;

insert into movies values(1, "ASSASSIN'S CREED: EMBERS", 'Animations'), (2, 'Real Steel', 'Animations'), (3, 'Alvin and the Chipmunks', 'Animations'), (4, 'The Adventures of Tin Tin', 'Animations'), (5, 'Safe', 'Action'), (6, 'Safe House', 'Action'), (7, 'GIA', '18+'), (8, 'Deadline 2009','18+'), (9, 'The Dirty Picture', '18+'), (10, 'Marley and me', 'Romance');

3. Create a table called authors with the following columns

authorid , name

- choose appropriate datatypes for the columns

a) Insert the following data into the table

1, J K Rowling

2, Thomas Hardy

3, Oscar Wilde

4, Sidney Sheldon

5, Alistair Maclean

6, Jane Autsen

b) Add a couple of authors of your choice

c) Change 'Alistair Maclean' to 'Alastair McNeal'

1. USE authors\_sql;

CREATE TABLE authors (authorid INT, name VARCHAR (30));

SELECT \* FROM authors\_sql.authors;

INSERT INTO authors\_sql.authors (authorid, name) VALUES (1, 'JK Rowling'), (2, 'Thomas Hardy'), (3, 'Oscar Wilde'), (4, 'Sidney Sheldon'), (5, 'Alistair Maclean'), (6, 'Jane Austen');

1. INSERT INTO authors (authorid, name) VALUES (7, 'Stan Lee'), (8, 'David Crane');
2. UPDATE authors SET name= 'Alastair McNeal' WHERE authorid=5;

4. Create a table called Books with the following columns

bookid, title, authorid

- choose appropriate datatypes for the columns

a) Insert the following records

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 'Random Book' from the table.

CREATE TABLE books (bookid INT, title VARCHAR (40), authorid INT);

1. USE authors\_sql;

CREATE TABLE books (bookid INT, title VARCHAR (40), authorid INT);

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 b WHERE b.title="Random Book";

5. Rename the table Books to Favbooks and Authors to Favauthors.

ALTER TABLE books RENAME To Favbooks;

ALTER TABLE authors RENAME TO Favauthors;

6. Create the following tables. Use auto increment wherever applicable

a. Products

product\_id - primary key

product\_name - cannot be null and only unique values are allowed

description

supplier\_id - foreign key of supplier table

CREATE TABLE products (

product\_id INT auto\_increment,

product\_name VARCHAR (40) NOT NULL UNIQUE,

supplier\_id INT,

PRIMARY KEY (product\_id),

FOREIGN KEY (supplier\_id) REFERENCES suppliers(supplier\_id)

);

b. Suppliers

supplier\_id - primary key

supplier\_name

location

CREATE TABLE suppliers (supplier\_id INT PRIMARY KEY, supplier\_name VARCHAR (30), location VARCHAR (40));

c. Stock

id - primary key

product\_id - foreign key of product table

balance\_stock

CREATE TABLE stock (

ID INT auto\_increment,

product\_id INT,

balance\_stock INT,

PRIMARY KEY (ID),

FOREIGN KEY (product\_id) REFERENCES products(product\_id)

);

7. Enter some records into the three tables.

insert into Suppliers (supplier\_name, location) values ('Haldirams', 'Nagpur'), ('PNG’, 'Pune'), ( 'Unilever', 'Mumbai');

insert into products values('P1', 'Namkeen', 'Snacks', 1);

insert into products values('P2','Necklace', 'Ornament', 2);

insert into products values('P3', 'Ring', 'Ornament', 2);

insert into products values('P4', 'Shampoo', 'FMCG', 3);

insert into products values('P5', 'Tooth paste', 'FMCG', 3);

insert into products values('P6', 'Sweets', 'Snacks’, 1);

insert into stock(product\_id, balance\_stock) values ('P3', 100);

insert into stock(product\_id, balance\_stock) values ('P1', 50);

insert into stock(product\_id, balance\_stock) values ('P6', 178);

insert into stock(product\_id, balance\_stock) values ('P2', 120);

insert into stock(product\_id, balance\_stock) values ('P4', 80);

insert into stock(product\_id, balance\_stock) values ('P6', 60);

8. Modify the supplier table to make supplier name unique and not null.

ALTER TABLE suppliers

MODIFY supplier\_name VARCHAR (40) UNIQUE NOT NULL;

9. Modify the emp table as follows

a. Add a column called deptno

b. Set the value of deptno in the following order

deptno = 20 where emp\_id is divisible by 2

deptno = 30 where emp\_id is divisible by 3

deptno = 40 where emp\_id is divisible by 4

deptno = 50 where emp\_id is divisible by 5

deptno = 10 for the remaining records.

SELECT \* FROM sql\_hr.employees;

UPDATE employees SET deptno=20 WHERE employee\_id%2=0;

UPDATE employees SET deptno=30 WHERE employee\_id%3=0;

UPDATE employees SET deptno=40 WHERE employee\_id%4=0;

UPDATE employees SET deptno=50 WHERE employee\_id%5=0;

UPDATE employees SET deptno=10 WHERE deptno is NULL;

10. Create a unique, hash index on the emp\_id column.

CREATE UNIQUE INDEX empid ON employee(emp\_id) USING HASH;

**Assignment 2**

1. select all employees in department 10 whose salary is greater than 3000. [table: employee]

SELECT\*

FROM employees

WHERE deptno=10 and salary>3000;

2. The grading of students based on the marks they have obtained is done as follows

40 to 50 -> Second Class

50 to 60 -> First Class

60 to 80 -> First Class

80 to 100 -> Distinctions

a. How many students have graduated with first class?

b. How many students have obtained distinction? [table: students]

a. alter table students add column grade varchar(50) not null;

update students set grade = "distinction" where marks between 80 and 100;

update students set grade = "fist class" where marks between 60 and 80;

update students set grade = "second class" where marks between 50 and 60;

update students set grade = "third class" where marks between 40 and 50;

select count (\*) from students where grade = "first class";

b. alter table students add column grade varchar(50) not null;

update students set grade = "distinction" where marks between 80 and 100;

update students set grade = "fist class" where marks between 60 and 80;

update students set grade = "second class" where marks between 50 and 60;

update students set grade = "third class" where marks between 40 and 50;

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

3. Get a list of city names from station with even ID numbers only. Exclude duplicates from your answer.[table: station]

SELECT \* FROM city WHERE id%2=0;

4. Find the difference between the total number of city entries in the table and the number of distinct city entries in the table. In other words, if N is the number of city entries in station, and N1 is the number of distinct city names in station, write a query to find the value of N-N1 from station.[table: station]

select count(city)-count(distinct city) from station;

select count(city) from station;

select count(distinct city) from station;

5. a. Query the list of CITY names starting with vowels (i.e., a, e, i, o, or u) from STATION. Your result cannot contain duplicates. [Hint: Use RIGHT() / LEFT() methods ]

b. Query the list of CITY names from STATION which have vowels (i.e., a, e, i, o, and u) as both their first and last characters. Your result cannot contain duplicates.

c. Query the list of CITY names from STATION that do not start with vowels. Your result cannot contain duplicates.

d. Query the list of CITY names from STATION that either do not start with vowels or do not end with vowels. Your result cannot contain duplicates. [table: station]

1. select distinct city from station where left(city,1) in ('a','e','i','o','u');
2. 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');
3. select distinct city from station where left(city,1) not in ('a','e','i','o','u');
4. 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. Write a query that prints a list of employee names having a salary greater than $2000 per month who have been employed for less than 10 months. Sort your result by ascending emp\_id. [table: emp]

select month(hire\_date), year(hire\_date) from emp where year(hire\_date)='2020';

select concat(first\_name," ",last\_name) as name,datediff(now(),hire\_date)/30 as months from emp where salary>2000 and datediff(now(),hire\_date)/30 <10 order by emp\_no;

8. 12. How much money does the company spend every month on salaries for each department? [table: emp]

Expected Result:

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

| sum(salary) | deptno |

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

| 30700.00 | 10 |

| 13600.00 | 20 |

| 2600.00 | 30 |

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

select sum(salary),deptno from employee group by deptno;

9. How many cities in the CITY table have a Population larger than 100000. [table: city]

select count(name) from city where population >100000;

10. What is the total population of California? [table: city]

select sum(population) from city where district='california';

11. What is the average population of the districts in each country? [table: city]

Expected Result:

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

| countrycode | avg(population) |

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

| JPN | 175839.2000 |

| NLD | 593321.0000 |

| USA | 120225.8750 |

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

select distinct (countrycode), avg (population) from city group by countrycode;

**Assignment 3**

Write a stored procedure that accepts the month and year as inputs and prints the ordernumber, orderdate and status of the orders placed in that month. The month should be abbreviated to three characters.

Example:

Input: month -> 'Feb'

year -> 2003

Output:

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

| orderdate | status |

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

| 2003-02-11 | Shipped |

| 2003-02-17 | Shipped |

| 2003-02-24 | Shipped |

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

3 rows in set (0.001 sec)

CREATE DEFINER=`root`@`localhost` PROCEDURE `new\_procedure`(mon varchar(3),yr varchar(4))

BEGIN

select orderdate , status from orders where orddate=concat(mon,yr);

END

FOR ADDING ADDITONAL COLUMN TO PAYMENTS TABLE (ORDATE—>2003—JAN)

select customernumber,purchase (customernumber) from payments where customernumber=114;

alter table orders add column orddate date;

insert into orders(orddate) values(date\_format(orderdate,"%M %Y"));

select concat( left(monthname(orderdate),3)," " ,year(orderdate)) as orddate from orders;

alter table orders modify column orddate varchar(50);

update orders set orddate=(concat( left(monthname(orderdate),3),year(orderdate)));

2. Write a stored procedure to insert a record into the cancellations table for all cancelled orders.

STEPS: a. Create a table called cancellations with the following fields

id (primary key), custumernumber (foreign key), ordernumber (foreign key), comments

All values except id should be taken from the order table.

Create table cancellations (id integer primary key auto\_increment,custnumber integer,ordnumber integer,foreign key(custnumber,ordnumber) references orders(customerNumber,orderNumber)

on delete cascade

on update cascade);

select ordernumber,customernumber,status from orders;

select customernumber,customername,purchase\_status(customernumber) from customers;

b. Read through the orders table. If an order is cancelled, then put an entry in the cancellations table.

BEGIN

declare id integer;

declare lcl\_ord integer;

declare lcl\_cust integer;

declare lcl\_st varchar(20);

declare can\_cur cursor for select ordernumber,customernumber,status from orders;

open can\_cur;

orderloop:loop

fetch can\_cur into lcl\_ord,lcl\_cust,lcl\_st;

if lcl\_st="cancelled" then

insert into cancellations values(id,lcl\_cust,lcl\_ord);

end if;

end loop orderloop;

close can\_cur;

END

3. a. Write function that takes the customernumber as input and returns the purchase\_status based on the following criteria. [table:Payments]

if the total purchase amount for the customer is < 25000 status = Silver, amount between 25000 and 50000, status = Gold

if amount > 50000 Platinum

CREATE DEFINER=`root`@`localhost` FUNCTION `purchase\_status`(cstno integer) RETURNS varchar(20) CHARSET utf8mb4

BEGIN

declare status varchar(20);

declare amt integer;

select sum(amount) into amt from payments where customernumber=cstno;

if (amt<25000) then

set status="silver";

elseif (amt between 25000 and 50000) then

set status="gold";

else

set status="platinum";

end if;

RETURN status;

END

b. Write a query that displays customerid, customername and purchase\_status

Select customernumber,customername,purchase\_status(customernumber) from customers;

4. Write a stored procedure that checks the creditlimit and the purchase status of the customers.

If a platinum customer has crediltlimit less than 100,000 raise an exception. In the exception handler update the crediltlimit to 100000.

If a silver customer has creditlimit greater than 60,000 raise an exception. In the exception handler update the crediltlimit to 60000.

CREATE PROCEDURE `check\_credit`()

BEGIN

declare finished, cnum integer default 0;

declare crlimit, uplimit decimal(10,2) default 0.0;

declare pstatus varchar(10) default '';

declare credit\_cur cursor for

select customerNumber, creditlimit, purchase\_status(customerNumber)

from customers

where purchase\_status(customerNumber) in ('platinum','gold');

declare exit handler for NOT FOUND SET finished = 1;

declare continue handler for SQLSTATE '45000'

BEGIN

Select concat("Updating ", cnum, " -> ", uplimit) as Message;

update customers

set creditlimit = uplimit

where customernumber = cnum;

END;

open credit\_cur;

creditloop: REPEAT

fetch credit\_cur into cnum, crlimit, pstatus;

if pstatus = 'platinum' and crlimit < 40000 then

set uplimit = 100000;

signal sqlstate '45000';

elseif pstatus = 'gold' and crlimit > 60000 then

set uplimit= 60000;

signal sqlstate '45000';

else

iterate creditloop;

end if ;

until finished = 1

end repeat creditloop;

END;

5. Replicate the functionality of 'on delete cascade' and 'on update cascade' using triggers on movies and rentals tables. Note: Both tables - movies and rentals - don't have primary or foreign keys. Use only triggers to implement the above.

CREATE DEFINER=`root`@`localhost` TRIGGER `movies\_BEFORE\_DELETE` BEFORE DELETE ON `movies` FOR EACH ROW BEGIN

delete from rentals where movieid = old.id;

END

CREATE DEFINER=`root`@`localhost` TRIGGER `movies\_BEFORE\_DELETE` BEFORE DELETE ON `movies` FOR EACH ROW

BEGIN

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

END