Sprint 5 Commands to be executed

SELECT Language -- outer SELECT expression

FROM CountryLanguage

WHERE CountryCode = ( -- left parenthesis - starts subquery

SELECT Code -- subquery SELECT expression

FROM Country

WHERE Name = 'Finland'

);

SELECT Country.Name, 100 \* Country.Population /

(SELECT SUM(Population) FROM Country) AS pct\_of\_world\_pop FROM Country;

SELECT Name,

(SELECT COUNT(\*) FROM City

WHERE CountryCode = Code) AS Cities,

(SELECT COUNT(\*) FROM CountryLanguage

WHERE CountryCode = Code) AS Languages

FROM Country;

-- Table Subquery

SELECT \* FROM ( SELECT Code, Name FROM Country

WHERE IndepYear IS NOT NULL ) AS IndependentCountries;

select avg(cont\_sum) from ( select Continent, sum(population) as cont\_sum from

country group by Continent) as t;

SELECT \* FROM City WHERE CountryCode IN (SELECT Code FROM Country WHERE Continent = 'Asia');

select \* from city where (CountryCode, Name) IN (select Code, Name from country where continent = 'Asia');

SELECT \* FROM City WHERE EXISTS (SELECT NULL FROM Country WHERE Capital = ID);

SELECT \* FROM Country WHERE NOT EXISTS (SELECT NULL FROM CountryLanguage WHERE CountryCode = Code AND Language = 'English');

SELECT 'Finland' = ANY (SELECT Name FROM world.Country);

SELECT \* FROM Country WHERE Population > ALL (SELECT Population FROM City);

SELECT \* FROM Country

WHERE Population > ANY (SELECT Population FROM City);

SELECT \* FROM Country

WHERE Population = SOME (SELECT Population FROM City);

SELECT \* FROM Country WHERE Population > (SELECT MIN(Population)  FROM City);

SELECT \* FROM Country WHERE Population < (SELECT MAX(Population)  FROM City);

SELECT \* FROM Country WHERE Population > (SELECT MAX(Population)  FROM City);

SELECT \* FROM Country WHERE Population < (SELECT MIN(Population)  FROM City);

SELECT \* FROM Country WHERE Population IN (SELECT Population FROM City);

SELECT \* FROM Country WHERE Population NOT IN (select population from city);

SELECT \* FROM City WHERE CountryCode IN (SELECT Code FROM Country WHERE Continent = 'Europe');

SELECT \* FROM Country WHERE NOT EXISTS (SELECT NULL FROM City WHERE CountryCode = Code);

SELECT \* FROM City WHERE CountryCode IN (SELECT Code FROM Country WHERE Name = 'Belgium');

SELECT \* FROM City WHERE CountryCode IN (SELECT Code FROM Country WHERE Country.Name = 'Belgium');

use sakila;

SELECT \* INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/MyFileOne.csv' FROM actor;

SELECT \* INTO OUTFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\MyFileTwo.csv' FROM actor;

(SELECT title, rental\_rate, release\_year from film)

INTO OUTFILE

'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\film.csv'

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n';

**Subquery Details**

CREATE DATABASE SUBQUERY\_Details;

USE SUBQUERY\_Details;

CREATE TABLE Library(

Book\_Number int NOT NULL,

Book\_Code varchar(15) NOT NULL,

Book\_Issue int NOT NULL,

Cost decimal(10, 2) NOT NULL,

PRIMARY KEY(Book\_Number, Book\_Code)

);

CREATE TABLE Book\_order(

Book\_Number int NOT NULL,

order\_Date date NOT NULL,

shipped\_Date date DEFAULT NULL,

Status varchar(50),

PRIMARY KEY(Book\_Number)

);

INSERT INTO Library( Book\_Number , Book\_Code , Book\_Issue , Cost )

VALUES(1001, 'A1000', 50, 55.02),

(1002, 'A1001', 50, 60.02),

(1003, 'A1002', 50, 60.02),

(1004, 'A1003', 50, 60.02),

(1005, 'A1004', 100, 60.02),

(1006, 'A1005', 100, 60.02),

(1007, 'A1006', 70, 60.02),

(1008, 'A1007', 70, 60.02),

(1009, 'A1008', 50, 60.02),

(1010, 'A1009', 50, 60.02);

INSERT INTO Book\_order( Book\_Number , order\_Date ,shipped\_Date , Status )

VALUES(1001, '2020-01-06', '2020-01-10', 'Done'),

(1002, '2020-01-11', '2020-01-15', 'Done'),

(1003, '2020-01-12', '2020-01-17', 'Done'),

(1004, '2020-05-06','2020-05-11','In progress'),

(1005, '2020-06-13', '2020-06-17', 'In progress'),

(1006, '2020-07-06', '2020-07-10', 'In progress'),

(1007, '2020-08-12', '2020-08-16', 'Done'),

(1008, '2020-09-25', '2020-09-30', 'Done'),

(1009, '2020-10-04', '2020-10-09', 'Done'),

(1010, '2020-11-06', '2020-11-12', 'Done');

-- Table subquery

SELECT book\_order.\*

FROM book\_order, library

WHERE book\_order.Book\_Number = library.Book\_Number

AND Status = 'In progress';

SELECT MAX(BI), MIN(BI), FLOOR(AVG(BI))

FROM(

SELECT book\_issue, count(book\_issue) AS BI FROM library GROUP BY book\_issue)

AS totalissue;

SELECT \* FROM book\_order bo, library l

WHERE bo.Book\_Number = l.Book\_Number

AND l.Book\_Issue = (

SELECT MAX(Book\_Issue) FROM library);

SELECT \* FROM book\_order bo, library l

WHERE bo.Book\_Number = l.Book\_Number

AND l.Book\_Issue = 100;

-- in

SELECT \* FROM book\_order WHERE Book\_Number IN(

SELECT book\_number FROM library );

SELECT \* FROM book\_order WHERE Book\_Number NOT IN(

SELECT book\_number FROM library);

/\* The comparison operator is used to compare Outer\_Query to the single query column value from each subquery result row. If we are using ALL clause then must match the all rows in subquery, or subquery must be empty. If we are using ANY or SOME clause, then must match at least one row in the subquery.\*/

SELECT \* FROM book\_order WHERE Book\_Number = ANY(

SELECT book\_number FROM library);

SELECT \* FROM book\_order WHERE Book\_Number = all(

SELECT book\_number FROM library);

/\* The EXISTS subquery is used to tests whether a sub-query returns at least one row or a qualifying row exists.\*/

SELECT \* FROM book\_order bo WHERE EXISTS(

SELECT \*

FROM library l WHERE bo.Book\_Number = l.Book\_Number);

/\* The EXISTS subquery is used to tests whether a sub-query returns at least one row or a qualifying row exists.

Any EXISTS subquery should contain an outer reference. It must be a correlated subquery.\*/

SELECT \* FROM book\_order bo WHERE NOT EXISTS(

SELECT \*

FROM library l WHERE bo.Book\_Number = l.Book\_Number);

use customerdetails;

SELECT Name

FROM employee

WHERE Emp\_ID = ANY

(SELECT Emp\_ID

FROM department

WHERE Dept\_No = 1001);

/\* The following SQL statement lists the Name from employee table if it finds ANY records in the department table has Dept\_no equal to 1001 (this will return TRUE because the Dept\_no column has some values of 1001):\*/

SELECT Name

FROM employee

WHERE Emp\_ID = ANY

(SELECT Emp\_ID

FROM department

WHERE Dept\_No > 1006);

/\*The following SQL statement lists the Name if it finds ANY records in the department table has Dept\_no larger than 1006 (this will return FALSE because the Dept\_no column has no values larger than 1006)\*/

SELECT ALL Name

FROM employee

WHERE TRUE;

/\* Displays all the names from employee\*/

SELECT Name

FROM Employee

WHERE Emp\_ID = ALL

(SELECT E\_ID

FROM department

WHERE Dept\_No = 1002);

/\* Here, query will not display all the names from the employee, it retrieves only

the matching record.\*/