```
-- This is a list of the SQL queries we executed in class

------------------------------------------------------------
-- SQL queries using one relation

-- What is the population of the US?
SELECT Population
FROM Country
WHERE Code = 'USA';

-- Which countries gained independence after 1989?
SELECT Name, IndepYear
FROM Country
WHERE IndepYear >= '1990';

-- Return all the attributes for cities with population over 1 million in the US
SELECT *
FROM City
WHERE Population >= '1000000' AND CountryCode = 'USA';

-- This query returns only the name of these cities, and renames the attribute
-- Name to LargeUSACity
SELECT Name AS LargeUSACity
FROM City
WHERE Population >= '1000000' AND CountryCode = 'USA';

-- Cities with population over 5 million (changing the unit to millions)
SELECT Name, (Population / 1000000) AS PopulationInMillion
FROM City
WHERE Population >= '5000000';

-- If we want to get a float value, and not the rounded integer value
SELECT Name, ((Population * 1.0)/ 1000000) AS PopulationInMillion
FROM City
WHERE Population >= '5000000';

-- Find the countries that have a form of goverment related to monarchy
SELECT Name, GovernmentForm
FROM Country
WHERE GovernmentForm LIKE '%Monarchy%';

-- Find all forms of goverment (the below query IS NOT CORRECT !!)
SELECT GovernmentForm
FROM Country ;

-- Hmmm, the above query keeps multiple copies of the same value.
-- Add the DISTINCT keyword to remove all the duplicates
SELECT DISTINCT GovernmentForm
FROM Country ;

-- We can use ORDER BY to order the city population by decreasing population
```

length : 19,678 lines : 691 Ln : 51 Col : 77 Sel : 0 | 0 Windows (CR LF) UTF-8 INS

Normal text file

```
-- We can use ORDER BY to order the city population by decreasing population
SELECT Name, (Population / 1000000) AS PopulationInMillion
FROM City
WHERE Population >= '5000000'
ORDER BY PopulationInMillion DESC;

-- Find the top two most populated cities!
-- Here we can use LIMIT to limit the output
SELECT Name, (Population / 1000000) AS PopulationInMillion
FROM City
ORDER BY PopulationInMillion DESC
LIMIT 2;

-- Which is the first country that became independent?
-- The query below is NOT CORRECT! NULL values are present
SELECT Name, IndepYear
FROM Country
ORDER BY IndepYear ASC
LIMIT 1;

-- Instead the correct way is to filter out the NULL values by using NOT NULL
SELECT Name, IndepYear
FROM Country
WHERE IndepYear NOT NULL
ORDER BY IndepYear ASC
LIMIT 1;

-----------------------------------------------------------------
-- Multi-relational queries

-- What are the names of all countries that speak Greek?
-- Here we need to join two relations
SELECT Name
FROM Country, CountryLanguage
WHERE Code = CountryCode AND Language = 'Greek';

-- Better style for the above query
SELECT Country.Name
FROM Country, CountryLanguage
WHERE Country.Code = CountryLanguage.CountryCode
AND CountryLanguage.Language = 'Greek';

-- or even better (prefer this one!)
SELECT C.Name
FROM Country C, CountryLanguage L
WHERE C.Code = L.CountryCode
AND L.Language = 'Greek';

-- Which countries speaks at least 50% Greek?
SELECT C.Name
FROM Country C, CountryLanguage L
```

```
SELECT C.Name
FROM Country C, CountryLanguage L
WHERE C.Code = L.CountryCode
AND L.Language = 'Greek'
AND L.Percentage >= 50 ;

-- What is the district of the capital of USA?
SELECT T.district
FROM Country C, City T
WHERE C.code = 'USA'
AND C.capital = T.id ;

-- Which countries speak Greek and English?
SELECT C.Name
FROM Country C, CountryLanguage L1, CountryLanguage L2
WHERE C.Code = L1.CountryCode
AND C.Code = L2.CountryCode
AND L1.Language = 'Greek'
AND L2.Language = 'English';

-- Alternatively we could also use the set operator INTERSECT
SELECT C.Name
FROM Country C, CountryLanguage L
WHERE C.Code = L.CountryCode
AND L.Language = 'Greek'
INTERSECT
SELECT C.Name
FROM Country C, CountryLanguage L
WHERE C.Code = L.CountryCode
AND L.Language = 'English' ;


-- Countries that have population more than 100 million and do not
-- speak English: here we use the set operator EXCEPT
SELECT C.Name
FROM Country C
WHERE C.Population >= 100000000
EXCEPT
SELECT C.Name
FROM Country C, CountryLanguage L
WHERE C.Code = L.CountryCode
AND L.Language = 'English' ;


------------------------------------------------------
-- Nested queries

-- In which country is Berlin?
SELECT C.Name
FROM Country C
WHERE C.code =
```

```
WHERE C.code =
 (SELECT C.CountryCode
  FROM City C
  WHERE C.name = 'Berlin');

-- Find all countries in Europe with population more than 50 million
SELECT C.Name
FROM (SELECT Name, Continent
     FROM Country
     WHERE Population >50000000) AS C
WHERE C.Continent = 'Europe' ;

-- Find all countries in Europe that have *some* city with population
-- more than 5 million
SELECT C.Name
FROM Country C
WHERE C.Continent = 'Europe'
AND C.Code IN (SELECT CountryCode
               FROM City
               WHERE Population > 5000000);

-- alternatively:
SELECT C.Name
FROM Country C
WHERE C.Continent = 'Europe'
AND EXISTS (SELECT *
            FROM City T
            WHERE T.Population > 5000000
            AND T.CountryCode = C.Code) ;

-- alternatively (the query below is NOT supported in SQLite!)
SELECT C.Name
FROM Country C
WHERE C.Continent = 'Europe'
AND 5000000 > ANY (SELECT T.Population
                   FROM City T
                   WHERE T.CountryCode = C.Code) ;

-- Find all countries in Europe that have all cities with population
-- less than 1 million
SELECT C.Name
FROM Country C
WHERE C.Continent = 'Europe'
AND NOT EXISTS (SELECT * FROM City T
                WHERE T.Population > 1000000
                AND T.CountryCode = C.Code) ;

-- alternatively (the query below is NOT supported by SQLite)
SELECT C.Name
FROM Country C
WHERE C.Continent = 'Europe'
```

```
-- alternatively (the query below is NOT supported by SQLite)
SELECT C.Name
FROM Country C
WHERE C.Continent = 'Europe'
AND 1000000 > ALL (SELECT T.Population
                   FROM City T
                   WHERE T.CountryCode = C.Code) ;



-------------------------------------------------------
-- Aggregate queries

-- Find the average population of countries in Europe
SELECT AVG(Population)
FROM Country
WHERE Continent = 'Europe';

-- Or max population:
SELECT MAX(Population)
FROM Country
WHERE Continent = 'Europe';

-- How many countries are in Europe?
SELECT COUNT(*)
FROM Country
WHERE Continent = 'Europe';

-- How many languages are spoken in the USA?
SELECT COUNT(*)
FROM CountryLanguage
WHERE CountryCode = 'USA';

-- Count the number of languages (this will NOT give the correct answer!!!)
SELECT COUNT(Language)
FROM CountryLanguage ;

-- Instead use DISTINCT to eliminate duplicates:
SELECT COUNT(DISTINCT Language)
FROM CountryLanguage ;

-- Find the name and population of the country with the max population in Europe!
-- The example below is not correct! Why?
SELECT Name, MAX(Population)
FROM Country
WHERE Continent = 'Europe';

-- Instead we have two alternatives:
-- # 1: Use subqueries to do this!
SELECT Name, Population
FROM Country
```

length : 19,678 lines : 691 Ln : 51 Col : 77 Sel : 0 | 0 Windows (CR LF) UTF-8 INS

Normal text file

```
-- Instead we have two alternatives:
-- # 1: Use subqueries to do this!
SELECT Name, Population
FROM Country
WHERE Population =
  (SELECT MAX(Population)
    FROM Country
    WHERE Continent = 'Europe');

-- # 1: Use order by and limit!
SELECT Name, Population
FROM Country
WHERE Continent = 'Europe'
ORDER BY Population DESC
LIMIT 1 ;

-- Find how many countries have each form of goverment:
SELECT GovernmentForm, COUNT(Code)
FROM Country
GROUP BY GovernmentForm ;

-- Find how many countries speak each language (with percentage > 50%) and
-- output them in decreasing order!
SELECT Language, COUNT(CountryCode) AS N
FROM CountryLanguage
WHERE Percentage >= 50
GROUP BY Language
ORDER BY N DESC ;

-- Find languages that are spoken in at least 3 different countries with
-- percentage at least 50
SELECT Language, COUNT(CountryCode) AS N
FROM CountryLanguage
WHERE Percentage >= 50
GROUP BY Language
HAVING N > 2
ORDER BY N DESC ;

-- The query below is NOT semantically correct!
-- HAVING clause is not correctly applied
SELECT Language, COUNT(CountryCode) AS N
FROM CountryLanguage
GROUP BY Language
HAVING Percentage > 50 ;

-- Output for each country the population most populated city,
-- for countries with at least 10 cities!
SELECT C.NAME AS Country, MAX(T.Population) AS N
FROM City T, Country C
WHERE C.Code = T.CountryCode
GROUP BY C.Name
```

```
-- Output for each country the population most populated city,
-- for countries with at least 10 cities!
SELECT C.NAME AS Country, MAX(T.Population) AS N
FROM City T, Country C
WHERE C.Code = T.CountryCode
GROUP BY C.Name
HAVING COUNT(T.ID) > 9
ORDER BY N DESC ;

-- How do we get the name of the most populated city as well?
SELECT Temp.Country, T.Name
FROM (SELECT C.NAME AS Country, C.Code AS Code, MAX(T.Population) AS N
      FROM City T, Country C
      WHERE C.Code = T.CountryCode
      GROUP BY C.Name, C.Code
      HAVING COUNT(T.ID) > 9) AS Temp, City T
WHERE T.Population = Temp.N
   AND T.CountryCode = Temp.Code;

-----------------------------------------------------------------
-- NULL behavior

SELECT COUNT(*)
FROM Country
WHERE IndepYear > 1990 OR IndepYear <= 1990 ;

-- Some countries are missing! What can we do?
SELECT COUNT(*)
FROM Country
WHERE IndepYear > 1990 OR IndepYear <= 1990 OR IndepYear IS NULL;


-- Returns max population of a city for each country, and for countries
-- without any city returns NULL
SELECT C.Name AS Country, MAX(T.Population) AS N
FROM Country C LEFT OUTER JOIN City T ON C.Code = T.CountryCode
GROUP BY C.Name
ORDER BY N DESC ;

-----------------------------------------------
-- DB Modifications

-- Inserting a new tuple!
INSERT INTO CountryLanguage VALUES('USA', 'C++', 'F', 0.5);

-- Let's delete it now.
DELETE FROM CountryLanguage WHERE LAnguage = 'C++';

-- Let's update something
UPDATE CountryLanguage
SET IsOfficial = 'T'
```

```
-- Let's update something
UPDATE CountryLanguage
SET IsOfficial = 'T'
WHERE CountryCode = 'USA' AND Language = 'Spanish';

-----------------------------------------------
-- Views

-- Let's create a view that stores the name, and the official languages for each country
CREATE VIEW OfficialCountryLanguage AS
SELECT C.Name AS CountryName, L.Language AS Language
FROM CountryLanguage L, Country C
WHERE L.CountryCode = C.Code
AND L.IsOfficial = 'T' ;

-- Now we can use this view! Find the countries with more than one offficial language!
SELECT CountryName, COUNT(Language)
FROM OfficialCountryLanguage
GROUP BY CountryName
HAVING COUNT(Language) > 1 ;

++++++++++++++++++++++++++++++++++++++++++++++++++

select date_sub(Now(), Interval 6 month);   //select 6 month old date
 //2018-03-13 15:09:12
SELECT * FROM test_users where user_created_date > date_sub(Now(), Interval 8 month);
--------------------------------
Table: employee_details
Columns:
EmpId int(11) PK
FullName varchar(45)
ManagerId varchar(45)
DateOfJoining varchar(4

Table: employee_salary
Columns:
EmpId int(11) PK
Project varchar(45)
Salary int(11)

-- get firstname of emp from fullname // ved prakash

SELECT SUBSTRING_INDEX(FullName,' ', 1) FROM employee_details; //ved

--------------------
-- Write a SQL query to fetch all the Employees who are also managers from EmployeeDetails table.

SELECT
    A.EmpId,
    A.FullName AS Emp,
    B.ManagerId,
```

```
-- Write a SQL query to fetch all the Employees who are also managers from EmployeeDetails table.

SELECT
    A.EmpId,
    A.FullName AS Emp,
    B.ManagerId,
    B.FullName AS Manager
FROM
    employee_details AS A
        JOIN
    employee_details AS B ON B.EmpId = A.ManagerId;
--------------------
-- Write a SQL query to fetch all employee records from EmployeeDetails table who have a salary record in EmployeeSalary table.

SELECT * FROM employee_details E
WHERE EXISTS
(SELECT * FROM employee_salary S WHERE  E.EmpId = S.EmpId);
----------------------
-- Write a SQL query to remove duplicates from a table without using temporary table.

DELETE FROM employee_salary
WHERE EmpId IN (
SELECT EmpId
FROM employee_salary
GROUP BY Project, Salary
HAVING COUNT(*) > 1);

---------------------------
-- Write a SQL query to fetch only odd rows from table.

SET @row_number = 0;
SELECT
    E.EmpId, E.Project, E.Salary
FROM
    (SELECT *,
        (@row_number:=@row_number + 1) AS RowNumber
    FROM
        employee_salary) E
WHERE
    E.RowNumber % 2 = 1;
------------------------------
-- Write a SQL query to fetch only even rows from table.

SET @row_number = 0;
SELECT
    E.EmpId, E.Project, E.Salary
FROM
    (SELECT *,
        (@row_number:=@row_number + 1) AS RowNumber
    FROM
        employee_salary) E
```

```
-- Write a SQL query to fetch only even rows from table.

SET @row_number = 0;
SELECT
    E.EmpId, E.Project, E.Salary
FROM
    (SELECT *,
        (@row_number:=@row_number + 1) AS RowNumber
    FROM
        employee_salary) E
WHERE
    E.RowNumber % 2 = 0;
------------------------------
-- Write a SQL query to create a new table with data and structure copied from another table.

create table Employee as select * from employee_details;
------------------------------
-- Write a SQL query to create an empty table with same structure as some other table.

create table Employee1 like employee_details;
------------------------------
mysql does not support intersect, minus

current datetime
select now()
------------------------------
-- Write a SQL query to fetch all the Employees from EmployeeDetails table who joined in Year 2016.

select * from employee_details where DateOfJoining between '01/01/2016' AND '31/12/2016';
------------------------------
-- Write a SQL query to fetch top n records?

select * from employee_details order by EmpId asc limit 6;
------------------------------
-- Write SQL query to find the nth highest salary from table. ex 5 th highest

select * from employee_salary order by Salary desc limit n-1, 1
------------------------------
-- Write SQL query to find the 5th highest salary from table without using TOP/limit keyword.

SELECT
    *
FROM
    employee_salary Emp1
WHERE
    4 = (SELECT
            COUNT(DISTINCT (Emp2.Salary))
        FROM
            employee_salary Emp2
        WHERE
            Emp2.Salary > Emp1.Salary)
```

```
-- Write SQL query to find the 5th highest salary from table without using TOP/limit keyword.

SELECT
   *
FROM
   employee_salary Emp1
WHERE
   4 = (SELECT
        COUNT(DISTINCT (Emp2.Salary))
      FROM
         employee_salary Emp2
      WHERE
         Emp2.Salary > Emp1.Salary)


SELECT *
FROM Employee Emp1
WHERE (N-1) = (
            SELECT COUNT(DISTINCT(Emp2.Salary))
            FROM Employee Emp2
            WHERE Emp2.Salary > Emp1.Salary
-------------------------------------
-- Select from table A which does not exist in table B

select A.*
from A left join B on A.BAND = B.HATE
where B.HATE IS NULL;
-------------------------------------
create table and give reference of foreign key in other table

CREATE TABLE Worker (
   WORKER_ID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
   FIRST_NAME CHAR(25),
   LAST_NAME CHAR(25),
   SALARY INT(15),
   JOINING_DATE DATETIME,
   DEPARTMENT CHAR(25)
);

CREATE TABLE Bonus (
   WORKER_REF_ID INT,
   BONUS_AMOUNT INT(10),
   BONUS_DATE DATETIME,
   FOREIGN KEY (WORKER_REF_ID)
      REFERENCES Worker(WORKER_ID)
      ON DELETE CASCADE
);

ON DELETE CASCADE -> It specifies that the child data is deleted when the parent data is deleted.
ON UPDATE (Optional) -> It specifies what to do with the child data when the parent data is updated.
```

```
ON DELETE CASCADE -> It specifies that the child data is deleted when the parent data is deleted.
ON UPDATE (Optional) -> It specifies what to do with the child data when the parent data is updated.

You have the options of NO ACTION(no changes in child data),
CASCADE((child data is deleted or updated also),
SET NULL(child data is set to Null),
or SET DEFAULT(child data is set to their default values).
------------------------------
Worker TAble with data
WORKER_ID  FIRST_NAME LAST_NAME   SALARY JOINING_DATE  DEPARTMENT
    1       Monika    Arora       100000 2/20/2014 9:00   HR
    2       Niharika  Verma       80000   6/11/2014 9:00 Admin
    3       Vishal    Singhal     300000 2/20/2014 9:00 HR
    4       Amitabh   Singh       500000 2/20/2014 9:00 Admin
    5       Vivek     Bhati    500000 6/11/2014 9:00 Admin
    6       Vipul     Diwan       200000 6/11/2014 9:00 Account
    7       Satish    Kumar       75000   1/20/2014 9:00 Account
    8       Geetika   Chauhan       90000   4/11/2014 9:00 Admin



//upper case
Select upper(FIRST_NAME) from Worker;

//first 3 character - substring
Select substring(FIRST_NAME,1,3) from Worker;

// check index of character: index of character a
Select INSTR(FIRST_NAME, BINARY'a') from Worker where FIRST_NAME = 'Amitabh';   // 5

//length of a field
Select distinct length(DEPARTMENT) from Worker;

//replace character
Select REPLACE(FIRST_NAME,'a','A') from Worker;

//concat
Select CONCAT(FIRST_NAME, ' ', LAST_NAME) AS 'COMPLETE_NAME' from Worker;

//end with h and contain 6 alphabets
Select * from Worker where FIRST_NAME like '_____h';

// year and month from date field , day, hour, minute,
Select * from Worker where year(JOINING_DATE) = 2014 and month(JOINING_DATE) = 2;

//show only odd rows from a table.
SELECT * FROM Worker WHERE MOD (WORKER_ID, 2) <> 0;

// even
SELECT * FROM Worker WHERE MOD (WORKER_ID, 2) = 0;
```

```
560  //clone a new table from another table
561  SELECT * INTO WorkerClone FROM Worker;
562  CREATE TABLE WorkerClone LIKE Worker;
563
564  //date
565  SELECT CURDATE();    //2019-05-21
566  SELECT NOW();   // 2019-05-21 17:59:36
567
568  //get employee with same salary
569  Select distinct W.WORKER_ID, W.FIRST_NAME, W.Salary
570  from Worker W, Worker W1
571  where W.Salary = W1.Salary
572  and W.WORKER_ID != W1.WORKER_ID;
573
574  //query to show one row twice in results from a table.
575  select FIRST_NAME, DEPARTMENT from worker W where W.DEPARTMENT='HR'
576  union all
577  select FIRST_NAME, DEPARTMENT from Worker W1 where W1.DEPARTMENT='HR';
578
579  // query to fetch the first 50% records from a table.
580  SELECT * FROM WORKER WHERE WORKER_ID <= (SELECT count(WORKER_ID)/2 from Worker);
581
582  //show the last record from a table.
583  Select * from Worker where WORKER_ID = (SELECT max(WORKER_ID) from Worker);
584
585  //select first row from table
586  Select * from Worker where WORKER_ID = (SELECT min(WORKER_ID) from Worker);
587
588  //SQL query to print the name of employees having the highest salary in each department.
589  SELECT
590     t.DEPARTMENT, t.FIRST_NAME, t.Salary
591  FROM
592     (SELECT
593        MAX(Salary) AS TotalSalary, DEPARTMENT
594     FROM
595        Worker
596     GROUP BY DEPARTMENT) AS TempNew
597        INNER JOIN
598     Worker t ON TempNew.DEPARTMENT = t.DEPARTMENT
599        AND TempNew.TotalSalary = t.Salary;
600
601
602  //get top 3 salary
603  SELECT DISTINCT
604     Salary
605  FROM
606     worker a
607  WHERE
608     3 >= (SELECT
609           COUNT(DISTINCT Salary)
610        FROM
```

Normal text file length : 19,678 lines : 691 Ln : 51 Col : 77 Sel : 0 | 0 Windows (CR LF) UTF-8 INS

```
//get top 3 salary
SELECT DISTINCT
    Salary
FROM
    worker a
WHERE
    3 >= (SELECT
            COUNT(DISTINCT Salary)
        FROM
            worker b
        WHERE
            a.Salary <= b.Salary)
ORDER BY a.Salary DESC;

// three min salaries from a table.
SELECT DISTINCT
    Salary
FROM
    worker a
WHERE
    3 >= (SELECT
            COUNT(DISTINCT Salary)
        FROM
            worker b
        WHERE
            a.Salary >= b.Salary)
ORDER BY a.Salary DESC;

//select name of workers with max salaries
SELECT FIRST_NAME, SALARY from Worker WHERE SALARY=(SELECT max(SALARY) from Worker);


Remove all duplicate row from table

with id in table
DELETE n1 FROM categories n1,
    categories n2
WHERE
    n1.id > n2.id AND n1.name = n2.name;

without id in table

-----------------------
Difference between Stored Procedure and Function in SQL Server::

Both stored procedures and functions are database objects which contain a set of SQL statements to complete a task
Stored Procedures are pre-compiled objects which are compiled for the first time and its compiled format is saved, which executes (compiled code) whenever it is called

A function is compiled and executed every time whenever it is called.

-----------------------
```

```
----------------------
Varchar vs varchar2 in oracle

VARCHAR is reserved by Oracle to support distinction between NULL and empty string .
VARCHAR2 does not distinguish between a NULL and empty string, and never will.
----------------------
Types  of indexes: use to speed up data retrieval

Index:
CREATE INDEX index_name ON table_name;

Single column index:
REATE INDEX index_name ON table_name (column_name);

Unique Indexes:
A unique index does not allow any duplicate values to be inserted into the table
CREATE UNIQUE INDEX index_name on table_name (column_name);

Composite Indexes:: on two or more columns of a table
CREATE INDEX index_name on table_name (column1, column2);

Drop index:
DROP INDEX index_name;
```