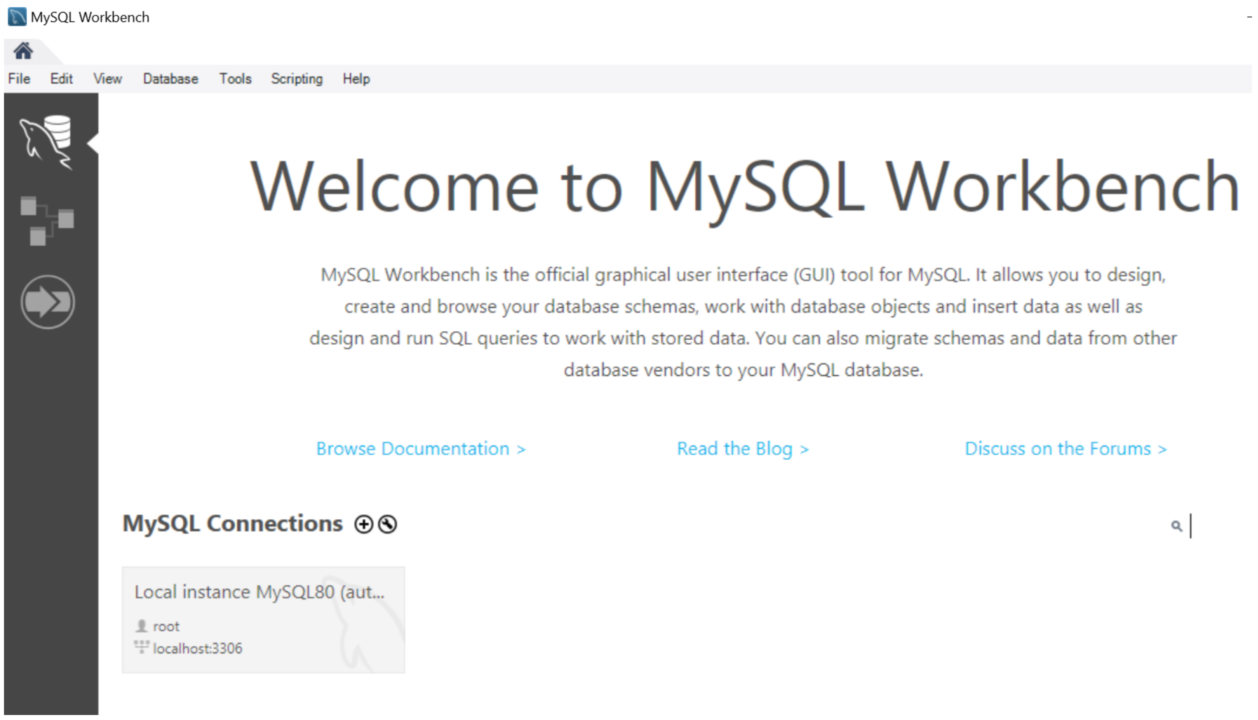
Data Analyst BootCamp

MySQL

Install SQL: Sql download link

Youtube Link: Youtube Link

MySQL Workbench:

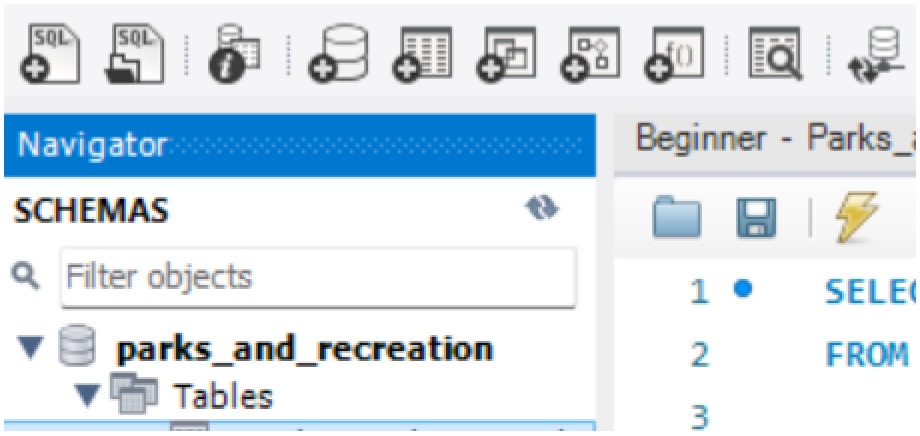


Welcome to MySQL Workbench

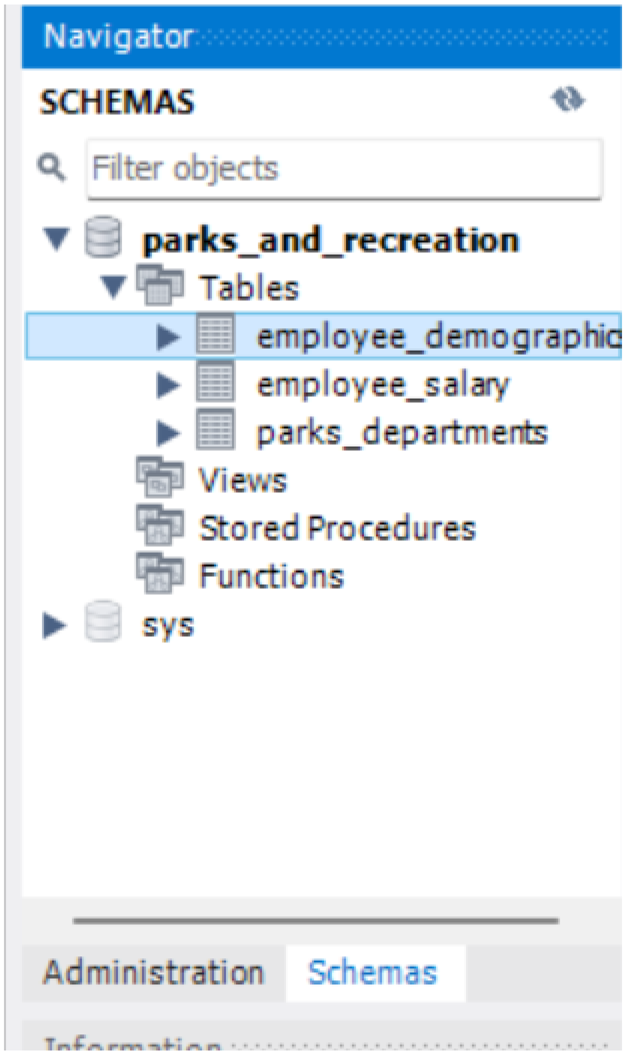
MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

Github Link to Download Data File: Parks_and_Rec_Create_db.sql

1.Open the file from your SQL(first row 2nd one)

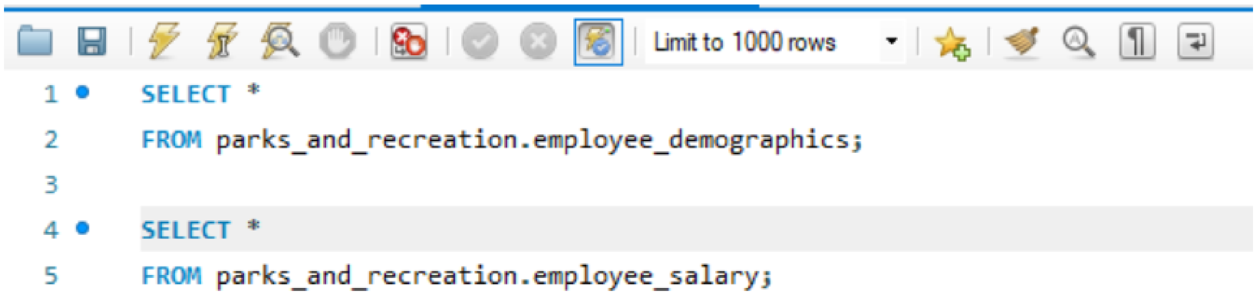


2.Refresh the schemas



3.Thunder bolt + I only exeucte that selected line

Thunder Bolt just will run all the commands



```sql
SELECT *
FROM parks_and_recreation.employee_demographics;

SELECT *
FROM parks_and_recreation.employee_salary;
```

4. Sample way how we created a DataBase

```sql
DROP DATABASE IF EXISTS `Parks_and_Recreation`;
CREATE DATABASE `Parks_and_Recreation`;
USE `Parks_and_Recreation`;



CREATE TABLE employee_demographics (
  employee_id INT NOT NULL,
  first_name VARCHAR(50),
  last_name VARCHAR(50),
  age INT,
  gender VARCHAR(10),
  birth_date DATE,
  PRIMARY KEY (employee_id)
);

CREATE TABLE employee_salary (
  employee_id INT NOT NULL,
  first_name VARCHAR(50) NOT NULL,
  last_name VARCHAR(50) NOT NULL,
  occupation VARCHAR(50),
  salary INT,
  dept_id INT
);
```

```
INSERT INTO employee_demographics (employee_id, first_name, last
VALUES
(1,'Leslie', 'Knope', 44, 'Female','1979-09-25'),
(3,'Tom', 'Haverford', 36, 'Male', '1987-03-04'),
(4, 'April', 'Ludgate', 29, 'Female', '1994-03-27'),
(5, 'Jerry', 'Gergich', 61, 'Male', '1962-08-28'),
(6, 'Donna', 'Meagle', 46, 'Female', '1977-07-30'),
(7, 'Ann', 'Perkins', 35, 'Female', '1988-12-01'),
(8, 'Chris', 'Traeger', 43, 'Male', '1980-11-11'),
(9, 'Ben', 'Wyatt', 38, 'Male', '1985-07-26'),
(10, 'Andy', 'Dwyer', 34, 'Male', '1989-03-25'),
(11, 'Mark', 'Brendanawicz', 40, 'Male', '1983-06-14'),
(12, 'Craig', 'Middlebrooks', 37, 'Male', '1986-07-27');


INSERT INTO employee_salary (employee_id, first_name, last_name,
VALUES
(1, 'Leslie', 'Knope', 'Deputy Director of Parks and Recreation
(2, 'Ron', 'Swanson', 'Director of Parks and Recreation', 70000,
(3, 'Tom', 'Haverford', 'Entrepreneur', 50000,1),
(4, 'April', 'Ludgate', 'Assistant to the Director of Parks and
(5, 'Jerry', 'Gergich', 'Office Manager', 50000,1),
(6, 'Donna', 'Meagle', 'Office Manager', 60000,1),
(7, 'Ann', 'Perkins', 'Nurse', 55000,4),
(8, 'Chris', 'Traeger', 'City Manager', 90000,3),
(9, 'Ben', 'Wyatt', 'State Auditor', 70000,6),
(10, 'Andy', 'Dwyer', 'Shoe Shiner and Musician', 20000, NULL),
(11, 'Mark', 'Brendanawicz', 'City Planner', 57000, 3),
(12, 'Craig', 'Middlebrooks', 'Parks Director', 65000,1);



CREATE TABLE parks_departments (
  department_id INT NOT NULL AUTO_INCREMENT,
  department_name varchar(50) NOT NULL,
  PRIMARY KEY (department_id)
);

INSERT INTO parks_departments (department_name)
VALUES
('Parks and Recreation'),
('Animal Control'),
('Public Works'),
('Healthcare'),
('Library'),
('Finance');
```

SELECT Statement: What columns you want to see in output

```
SELECT *
FROM parks_and_recreation.employee_demographics;
#The above one for selecting all columns in db

SELECT first_name
FROM parks_and_recreation.employee_demographics;
#This will select only first_name column in db

SELECT first_name,last_name,gender
FROM parks_and_recreation.employee_demographics;
#This will select  multiple column in db

SELECT first_name,
last_name,
gender,
age+10
FROM parks_and_recreation.employee_demographics;
#This will use for calculations purpose
#PEMDAS order ( Paranthesis,Exponential,Multiplication,Divison,
SELECT DISTINCT gender
FROM parks_and_recreation.employee_demographics;
#This Distinct will select only things single time which is most
```

WHERE Statement: filter our data

```
SELECT *
FROM parks_and_recreation.employee_demographics
WHERE first_name='Leslie';
#To select only Leslie data

SELECT *
FROM parks_and_recreation.employee_salary
WHERE salary>=50000;

SELECT *
FROM parks_and_recreation.employee_demographics
WHERE gender !='female';
#Just using statements as per our convinence

SELECT *
FROM parks_and_recreation.employee_demographics
WHERE birth_date > '1947-04-22';
#YYYY-MM-DD format standard

#Logical Operators (AND OR NOT)
SELECT *
FROM parks_and_recreation.employee_demographics
WHERE birth_date > '1947-04-22'
AND gender='male';

SELECT *
FROM parks_and_recreation.employee_demographics
```

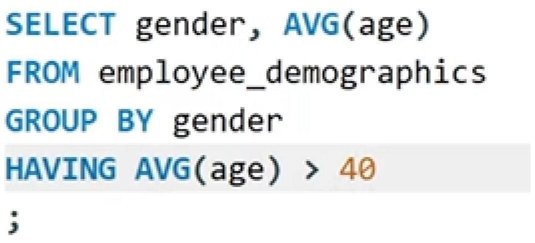
```
WHERE first_name like 'a%';
#gives all names starting with a
#Adny, Akash, Ariyan
SELECT *
FROM parks_and_recreation.employee_demographics
WHERE first_name like '%ar%';
#gives all names starting with before something ar after somethi
#Mark

SELECT *
FROM parks_and_recreation.employee_demographics
WHERE first_name like 'a__';
#gives all names starting with a exact match two characters afte
#Ann
```

GROUP BY and ORDER BY in SQL:

```
SELECT gender,AVG(age),MAX(age),MIN(age),COUNT(age)
FROM parks_and_recreation.employee_demographics
GROUP BY gender;
#we will get the avg age of male and female by grouping and usi

SELECT occupation,salary
FROM employee_salary
GROUP BY occupation,salary;
#To get occcupations and their salaries

#ORDERBY
SELECT *
FROM employee_demographics
ORDER BY first_name DESC;
#by default ASC we can change to DESC order well by changing thi
```

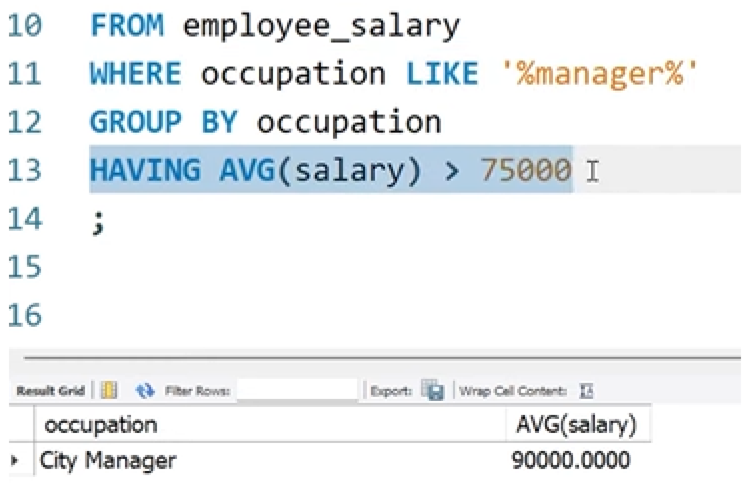
```
SELECT *
FROM employee_demographics
ORDER BY gender,age;
#firstly it will order by Female and male and in that also less
#can change to age desc by ORDER BY gender,age DESC;gender will
#Not every time u can use colmun name u can use numbering like (
```

HAVING vs WHERE in MySQL:



```
SELECT gender, AVG(age)
FROM employee_demographics
GROUP BY gender
HAVING AVG(age) > 40
;
```

We cant directly put it in before group BY we have to wait since it has to be created



```
FROM employee_salary
WHERE occupation LIKE '%manager%'
GROUP BY occupation
HAVING AVG(salary) > 75000
;
```

```
occupation      AVG(salary)
City Manager    90000.0000
```

Filter manager and as well as avg(salary) both

Having works for aggregate functions

```
#Limit and Aliasing
SELECT *
FROM employee_demographics
LIMIT 3;
#limited to top3

SELECT *
FROM employee_demographics
ORDER BY age DESC
LIMIT 3;
#we can use like this which can be powerful
#LIMIT 2,1 means we will start at second row we will go one row

#Aliasing
SELECT gender,avg(age) AS avg_age
FROM employee_demographics
GROUP BY gender
HAVING avg_age>40
#we can do like this
```

JOINS in MySQL

```
select *
FROM employee_demographics;

select *
FROM employee_salary;

select *
FROM employee_demographics
INNER JOIN employee_salary
     ON employee_demographics.employee_id=employee_salary.employ
;
```

```
select *
FROM employee_demographics AS dem
INNER JOIN employee_salary AS sal
     ON dem.employee_id=sal.employee_id
;

select dem.employee_id,age,occupation
FROM employee_demographics AS dem
INNER JOIN employee_salary AS sal
     ON dem.employee_id=sal.employee_id
;
#outer join consists of left and right diff is it takes everythi
#Left join : everything from left table matches with right table
#Right join :everything from right table matches with left table
select *
FROM employee_demographics AS dem
RIGHT JOIN employee_salary AS sal
     ON dem.employee_id=sal.employee_id
;

#self join
#just for assigning
SELECT emp1.employee_id,emp1.first_name,emp1.last_name,emp2.empl
FROM employee_salary emp1
JOIN employee_salary emp2
    ON emp1.employee_id + 1  = emp2.employee_id
;

#joining multiple tables together
select *
FROM employee_demographics AS dem
INNER JOIN employee_salary AS sal
     ON dem.employee_id=sal.employee_id
INNER JOIN parks_departments AS pd
     ON sal.dept_id=pd.department_id
;
```

```
#here 1st two tables we are having one common column
#where as in the 2nd two tables we are having another column cor

select *
from employee_salary
```

UNIONS in MySQL

```
#Union allows you to combine rows from sep/same tables
#Joins allows you to combine columns from sep/same tables
SELECT first_name, last_name, 'Old Lady' as Label
FROM employee_demographics
WHERE age > 40 AND gender = 'Female'
UNION
SELECT first_name, last_name, 'Old Man'
FROM employee_demographics
WHERE age > 40 AND gender = 'Male'
UNION
SELECT first_name, last_name, 'Highly Paid Employee'
FROM employee_salary
WHERE salary >= 70000
ORDER BY first_name
;
```

String Functions:

```
#String functions

#random thing to check lenght
select length('Manoj');

#from table just checking length of first name
```

```
select first_name,length(first_name)
from employee_demographics
order by 2;
#order by 1 means we are seeing 1st column
#order by 2 means we are seeing 2nd column and sorting as per th

select upper('sky');
select lower('SKY');

select first_name,UPPER(first_name)
from employee_demographics;

select TRIM('      SKY     ');#trim total left and right space
select RTRIM('      sky     ');#right trim
select LTRIM('      sky      ');#left trim

select first_name,
left(first_name,4),
right(first_name,4),
substring(first_name,3,2),
substring(birth_date,6,2) as birthmonth
from employee_demographics;

select first_name,replace(first_name,'a','z')
from employee_demographics;

select locate('x','Alexander');

select first_name,last_name,
concat(first_name,' ',last_name) as Full_name
from employee_demographics;
```

Case statements in MySQL

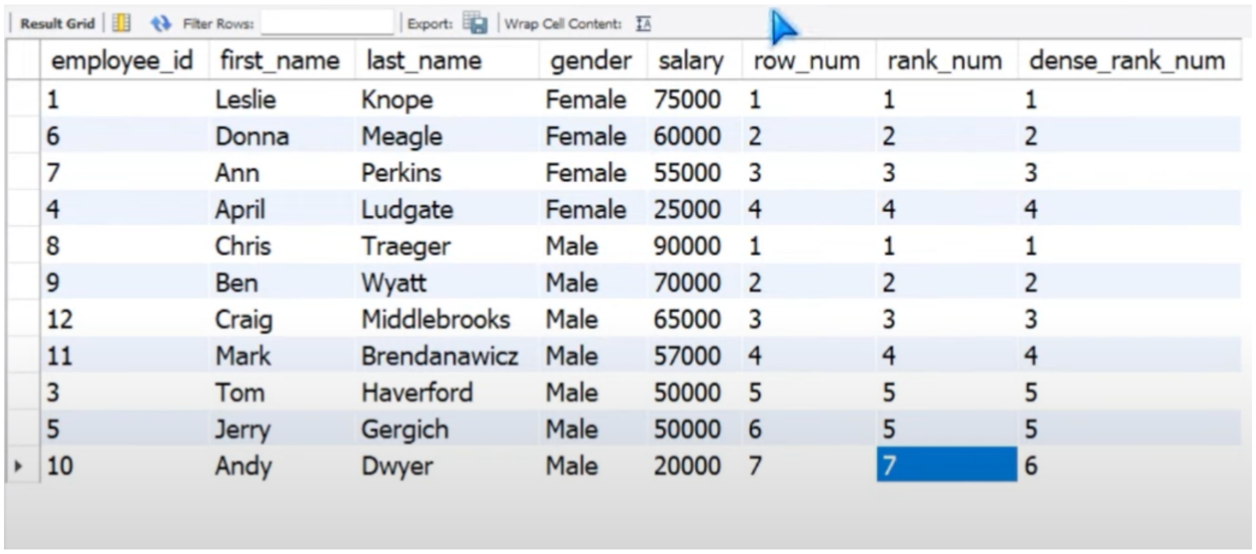
```
#case statements
select first_name,last_name,
age,
CASE
    when age<=30 then 'young'
    when age between 31 and 50 then 'old'
END as agebracket
from employee_demographics;

-- payincrease and bonus
-- if salary<50000 5%
-- if salary>50000 7%
-- if in finance then 10%bonus

select first_name,
last_name,
salary,
case
    when salary<=50000 then salary+(salary*0.05)
    when salary>50000 then salary+(salary*0.07)
end as new_salary,
case
    when dept_id=6 then salary+(salary*0.10)
end as bonus
from employee_salary;


select *
from employee_salary;

select *
from parks_departments;
```

Subqueries in MySQL

```
SELECT *
FROM employee_demographics
WHERE employee_id IN
            (SELECT employee_id
                FROM employee_salary
                WHERE dept_id = 1);



SELECT first_name,
salary,
(SELECT AVG(salary)
    FROM employee_salary) as avg_salary
FROM employee_salary;
```

Window functions in MySQL

```
-- window functions
select gender, avg(salary) as avg_salary
from employee_demographics as dem
join employee_salary as sal
  on dem.employee_id=sal.employee_id
group by gender;

-- using window function
select dem.first_name,dem.last_name,avg(salary) over(partition b
from employee_demographics as dem
join employee_salary as sal
  on dem.employee_id=sal.employee_id;

  -- rolling sum function most important
select dem.first_name,dem.last_name,gender,salary,
sum(salary) over(partition by gender order by dem.employee_id) a
from employee_demographics as dem
join employee_salary as sal
  on dem.employee_id=sal.employee_id;
  -- general things most important
select dem.first_name,dem.last_name,gender,salary,
row_number() over(partition by gender order by salary desc),
rank() over(partition by gender order by salary desc),
dense_rank() over(partition by gender order by salary desc)
from employee_demographics as dem
join employee_salary as sal
  on dem.employee_id=sal.employee_id;
```



employee_id	first_name	last_name	gender	salary	row_num	rank_num	dense_rank_num
1	Leslie	Knope	Female	75000	1	1	1
6	Donna	Meagle	Female	60000	2	2	2
7	Ann	Perkins	Female	55000	3	3	3
4	April	Ludgate	Female	25000	4	4	4
8	Chris	Traeger	Male	90000	1	1	1
9	Ben	Wyatt	Male	70000	2	2	2
12	Craig	Middlebrooks	Male	65000	3	3	3
11	Mark	Brendanawicz	Male	57000	4	4	4
3	Tom	Haverford	Male	50000	5	5	5
5	Jerry	Gergich	Male	50000	6	5	5
10	Andy	Dwyer	Male	20000	7	7	6

CTEs(Common Table Expression) in MySQL

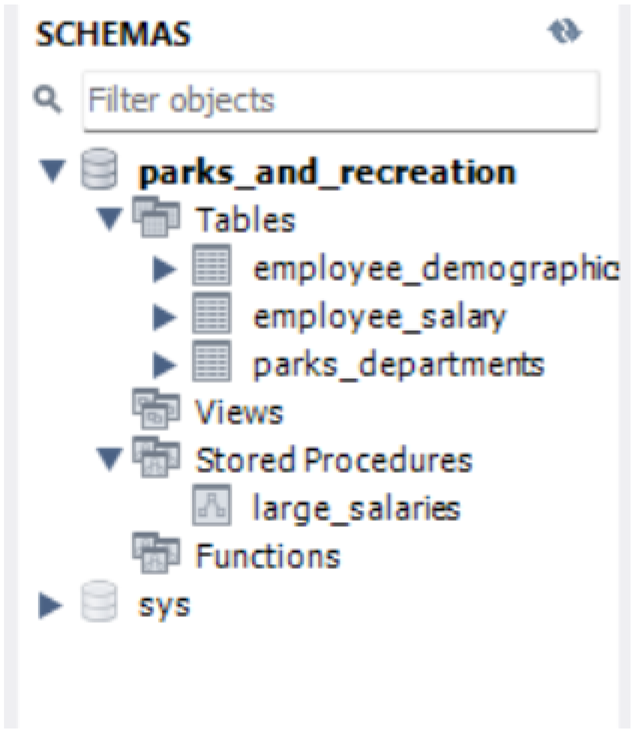
```
WITH CTE_Example AS
(
SELECT gender, SUM(salary) as sum_sal, MIN(salary) as min_sal, M
FROM employee_demographics dem
JOIN employee_salary sal
    ON dem.employee_id = sal.employee_id
GROUP BY gender
)
```

```
-- directly after using it we can query the CTE
SELECT avg(avg_sal)
FROM CTE_Example;



WITH CTE_Example AS
(
SELECT employee_id, gender, birth_date
FROM employee_demographics dem
WHERE birth_date > '1985-01-01'
), -- just have to separate by using a comma
CTE_Example2 AS
(
SELECT employee_id, salary
FROM parks_and_recreation.employee_salary
WHERE salary >= 50000
)
-- Now if we change this a bit, we can join these two CTEs toget
SELECT *
FROM CTE_Example cte1
LEFT JOIN CTE_Example2 cte2
    ON cte1. employee_id = cte2. employee_id;
```

Temp Tables in MySQL

```
-- Temporary Tables
CREATE TEMPORARY TABLE temp_table
(first_name varchar(50),
last_name varchar(50),
favorite_movie varchar(100)
);

SELECT *
FROM temp_table;
-- notice that if we refresh out tables it isn't there. It isn'
-- now obviously it's balnk so we would need to insert data int

INSERT INTO temp_table
VALUES ('Alex','Freberg','Lord of the Rings: The Twin Towers');

-- now when we run it and execute it again we have our data
SELECT *
FROM temp_table;

-- another way of creating a temp table
CREATE TEMPORARY TABLE salary_over_50k
SELECT *
FROM employee_salary
WHERE salary > 50000;

-- if we run this query we get our output
SELECT *
FROM salary_over_50k;
```

Stored Procedures in MySQL



```
-- Stored procedures in mysql

CREATE PROCEDURE large_salaries()
SELECT *
FROM employee_salary
WHERE salary>=50000;
-- side schema u have created a new stored procedures
CALL large_salaries();

-- two conditions implied
DELIMITER $$
CREATE PROCEDURE large_salaries2()
BEGIN
    SELECT *
    FROM employee_salary
    WHERE salary >= 60000;
    SELECT *
    FROM employee_salary
    WHERE salary >= 50000;
```

```
END $$

-- now we change the delimiter back after we use it to make it c
DELIMITER ;

Call large_salaries2();

-- with help of parameter we are solving
DELIMITER $$
CREATE PROCEDURE large_salaries3(employee_id_param INT)
BEGIN
    SELECT salary
    FROM employee_salary
    WHERE employee_id = employee_id_param;
END $$

DELIMITER ;



CALL large_salaries3(7);
```

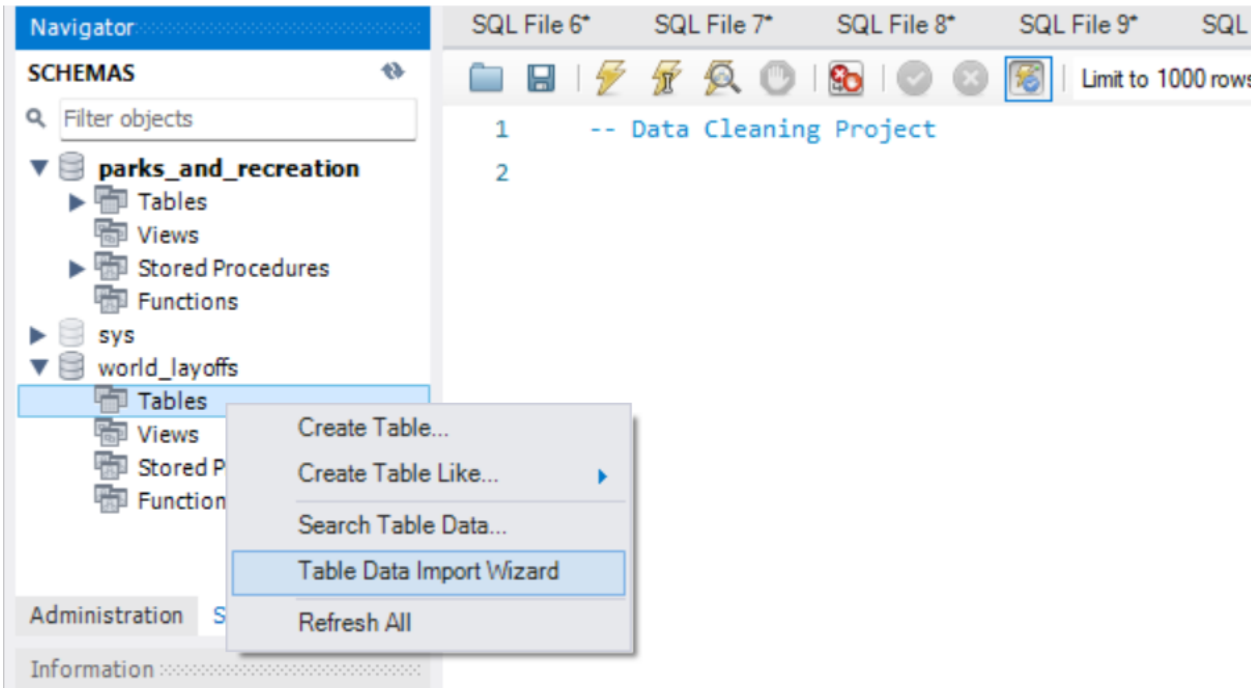
Triggers and Events in MySQL

```
-- Triggers

-- a Trigger is a block of code that executes automatically exec

-- for example we have these 2 tables, invoice and payments - wh
-- to reflect that the client has indeed paid their invoice



SELECT * FROM employee_salary;
SELECT * FROM employee_demographics;
-- so really when we get a new row or data is inserted into the
-- with the amount that was paid
-- so let's write this out
USE parks_and_recreation;
DELIMITER $$

CREATE TRIGGER employee_insert2
    -- we can also do BEFORE, but for this lesson we have to do
    AFTER INSERT ON employee_salary
    -- now this means this trigger gets activated for each row t
    -- only trigger once, but MySQL doesn't have this functional
    FOR EACH ROW

    -- now we can write our block of code that we want to run wh
BEGIN
-- we want to update our client invoices table
-- and set the total paid = total_paid (if they had already made
-- NEW says only from the new rows that were inserted. There is
    INSERT INTO employee_demographics (employee_id, first_name,
END $$

DELIMITER ;

-- Now let's run it and create it



-- Now that it's created let's test it out.

-- Let's insert a payment into the payments table and see if it

-- so let's put the values that we want to insert - let's pay of
INSERT INTO employee_salary (employee_id, first_name, last_name,
VALUES(13, 'Jean-Ralphio', 'Saperstein', 'Entertainment 720 CEO
-- now it was updated in the payments table and the trigger was



-- Events
SELECT *
```

```
FROM parks_and_recreation.employee_demographics;

SHOW EVENTS;

-- we can drop or alter these events like this:
DROP EVENT IF EXISTS delete_retirees;
DELIMITER $$
CREATE EVENT delete_retirees
ON SCHEDULE EVERY 30 SECOND
DO BEGIN
    DELETE
    FROM parks_and_recreation.employee_demographics
    WHERE age >= 60;
END $$


-- if we run it again you can see Jerry is now fired -- or I mea
SELECT *
FROM parks_and_recreation.employee_demographics;
```

Data Cleaning Project

Firstly create a new schema with name world_layoffs then do this



```
-- Data Cleaning Project
```

Then choose the layofs.csv file

```
-- Data Cleaning Project

-- checking the imported data
select *
from layoffs;

-- step1 remove duplicates
-- step2 standardize data
-- step3 null values or blank values
-- step4 remove anycolumns not necessary

-- creating new table because we want backup data

create table layoffs_staging
like layoffs;

select *
from layoffs_staging;
```

```
insert layoffs_staging
select *
from layoffs;

select *,
row_number() over(partition by company,industry,total_laid_off,p
from layoffs_staging;

with duplicate_cte as
(
    select *,
    row_number() over(partition by company,location,industry,tot
    from layoffs_staging
)
select *
from duplicate_cte
where rownumber>1;

-- to confirm are duplicates?
select *
from layoffs_staging
where company='Elemy';

-- partition by everything

CREATE TABLE `layoffs_staging2` (
  `company` text,
  `location` text,
  `industry` text,
  `total_laid_off` int DEFAULT NULL,
  `percentage_laid_off` text,
  `date` text,
  `stage` text,
  `country` text,
  `funds_raised_millions` int DEFAULT NULL,
  `rownumber` INT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_

select *
from layoffs_staging2
where rownumber>1;

insert into layoffs_staging2
select *,
    row_number() over(partition by company,location,industry,tot
    from layoffs_staging;

delete
from layoffs_staging2
where rownumber>1;

select *
from layoffs_staging2;

-- standardizing data
select company,trim(company)
from layoffs_staging2;

-- trim takes the white space at beginning and end
update layoffs_staging2
set company= trim(company);

select *
from layoffs_staging2
where industry like 'crypto%';

update layoffs_staging2
set industry='Crypto'
where industry like 'crypto%';

select distinct industry
```

```
from layoffs_staging2
order by 1;

-- unitedstates and unitedstates. issue fixing
select distinct country, trim(trailing'.'from country)
from layoffs_staging2
order by 1;

update layoffs_staging2
set country=trim(trailing'.'from country)
where country like 'United States%';

select *
from layoffs_staging2;

-- for date text is not good as we have to do time series
select `date`,
STR_TO_DATE(`date`,'%m/%d/%Y') -- take care of m,d,Y it should 
from layoffs_staging2;


update layoffs_staging2
set `date`=STR_TO_DATE(`date`,'%m/%d/%Y');

select *
from layoffs_staging2;

-- changing date type to date rather than a text
Alter table layoffs_staging2
modify column `date` DATE;

-- step3 null and blank values


update layoffs_staging2
set industry =null
```

```
where industry='';

update layoffs_staging2 t1
join layoffs_staging2 t2
    on t1.company=t2.company
set t1.industry=t2.industry
where t1.industry is null
and t2.industry is not null;

select *
from layoffs_staging2;

-- when u r confident enough delete then only
-- here below both are null so no use so we can delete it
select *
from layoffs_staging2
where total_laid_off is  null
and percentage_laid_off is null;

delete
from layoffs_staging2
where total_laid_off is  null
and percentage_laid_off is null;

select *
from layoffs_staging2;

-- drop row number column
alter table layoffs_staging2
drop column rownumber;

select *
from layoffs_staging2;
```

MySQL Exploratory Data Analysis

```
-- exploratory data analysis

select *
from layoffs_staging2;

-- checking max of totallaidoff and percentagelaidoff
select max(total_laid_off),max(percentage_laid_off)
from layoffs_staging2;


-- checking percentagelaidoff and totallaidoff
select *
from layoffs_staging2
where percentage_laid_off=1
order by total_laid_off desc;

-- checking max of totallaidoff and percentagelaidoff
select max(total_laid_off),max(percentage_laid_off)
from layoffs_staging2;


-- checking percentagelaidoff and fundsraisedmillions
select *
from layoffs_staging2
where percentage_laid_off=1
order by funds_raised_millions desc;

-- group by
select company,sum(total_laid_off)
from layoffs_staging2
group by company
order by 2 desc; -- order by which column which order

-- date range
select min(`date`),max(`date`)
```

```
from layoffs_staging2;

-- check which industry
select industry,sum(total_laid_off)
from layoffs_staging2
group by industry
order by 2 desc;

-- check which country
select country,sum(total_laid_off)
from layoffs_staging2
group by country
order by 2 desc;

-- check with date
select `date`,sum(total_laid_off)
from layoffs_staging2
group by `date`
order by 1 desc;

-- check with year
select year(`date`),sum(total_laid_off)
from layoffs_staging2
group by year(`date`)
order by 1 desc;

-- check with stage
select stage,sum(total_laid_off)
from layoffs_staging2
group by stage
order by 2 desc;

-- checking month and year wise
select substring(`date`,1,7) as `Month`,sum(total_laid_off)
from layoffs_staging2
where substring(`date`,1,7) is not null
```

```
group by `month`
order by 1 asc;

-- rolling total
with rolling_total as
(
select substring(`date`,1,7) as `Month`,sum(total_laid_off) as t
from layoffs_staging2
where substring(`date`,1,7) is not null
group by `month`
order by 1 asc
)
select `month`,total_off, sum(total_off) over(order by `month`)
from rolling_total;

-- checking company wise analysis
select company,year(`date`),sum(total_laid_off)
from layoffs_staging2
group by company,year(`date`)
order by sum(total_laid_off) desc;

-- some analysis
WITH Company_Year AS
(
  SELECT company, YEAR(date) AS years, SUM(total_laid_off) AS to
  FROM layoffs_staging2
  GROUP BY company, YEAR(date)
)
, Company_Year_Rank AS (
  SELECT company, years, total_laid_off, DENSE_RANK() OVER (PART
  FROM Company_Year
)
SELECT company, years, total_laid_off, ranking
FROM Company_Year_Rank
WHERE ranking <= 3
```

```
AND years IS NOT NULL
ORDER BY years ASC, total_laid_off DESC;
```