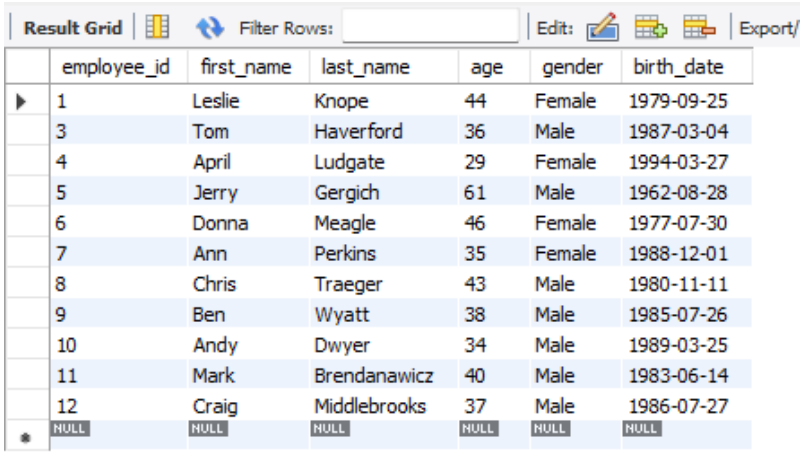
SQL Beginner Series Notes

This is the full fable we are going to use for this note:



Result Grid | Filter Rows: | Edit: | Export/

employee_id	first_name	last_name	age	gender	birth_date
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
NULL	NULL	NULL	NULL	NULL	NULL

Default structure:

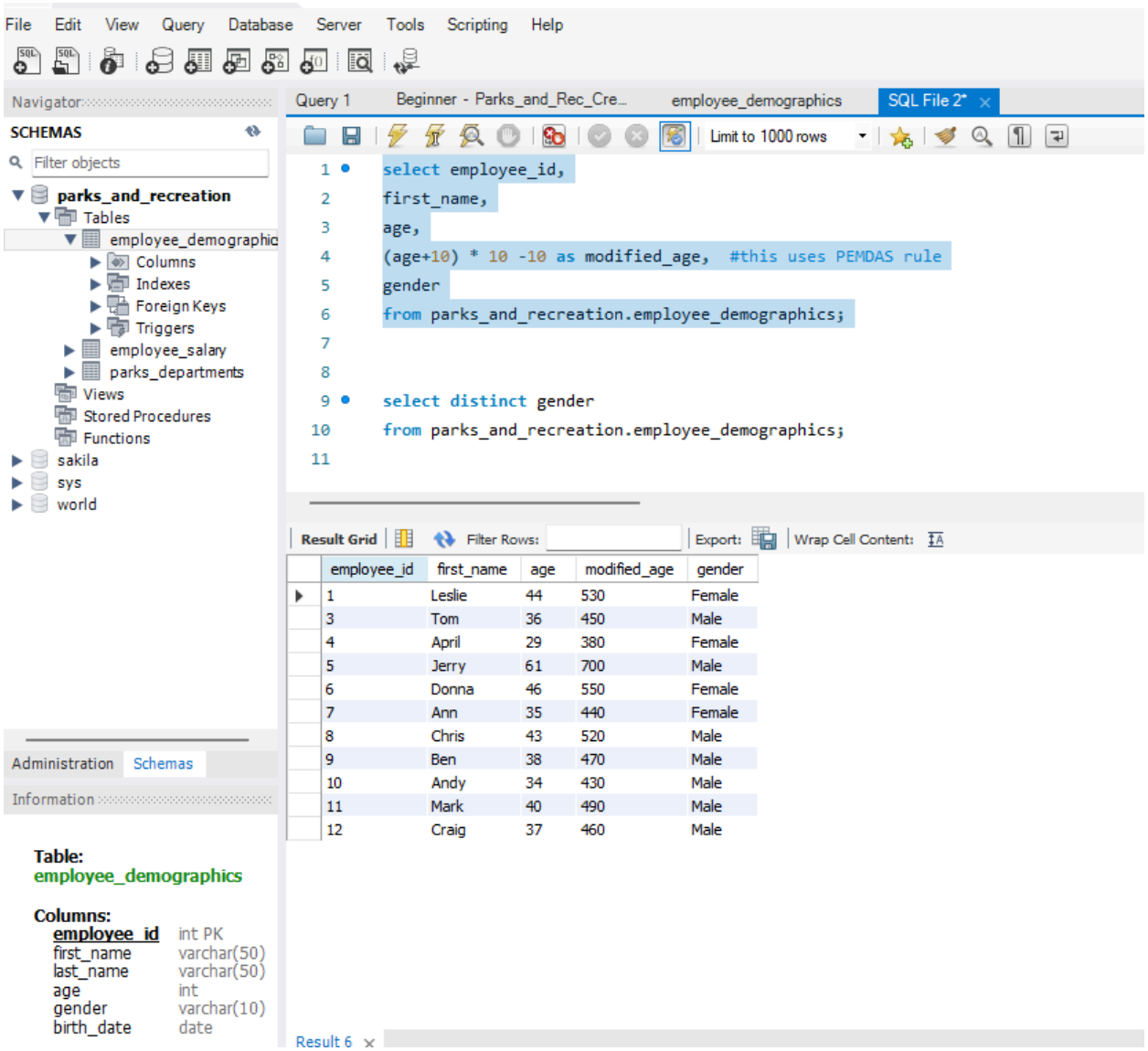
Select ____
From _____
Where ____
Group by _____
Having _____

1)Select

Selecting some columns from parks_and_recreation databases employee_demographics file

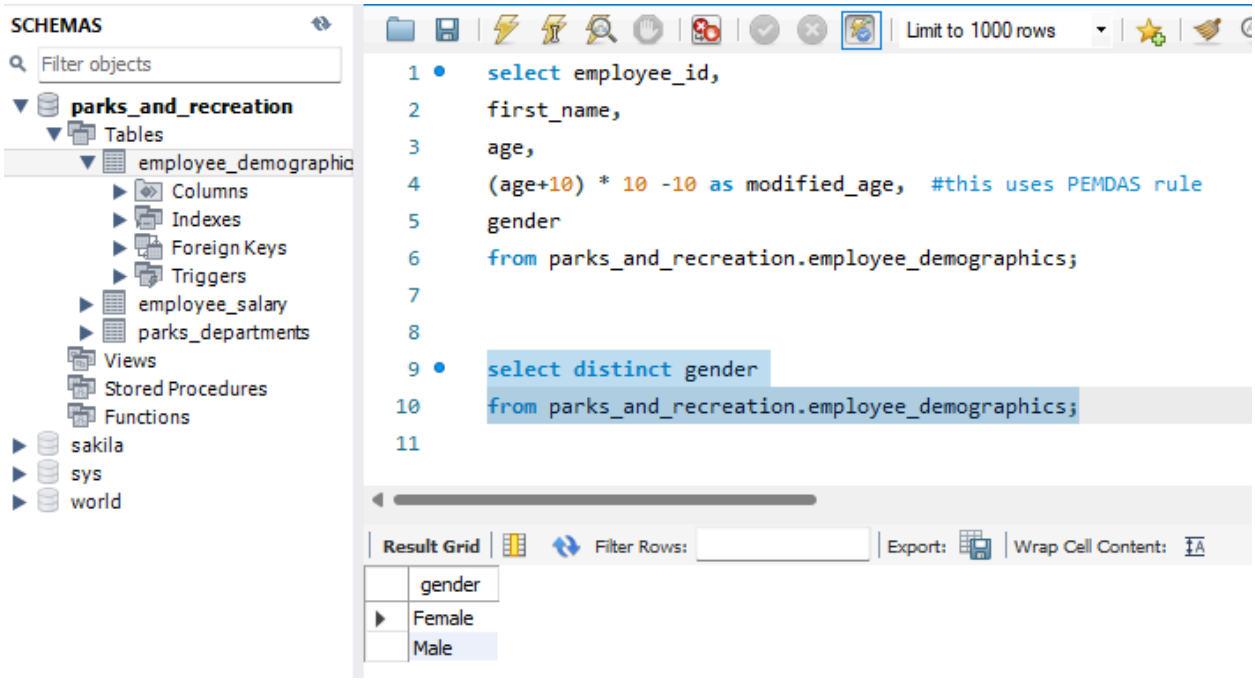
Query:

```
select employee_id,
first_name,
age,
(age+10) * 10 -10 as modified_age,  #this uses PEMDAS rule
gender
from parks_and_recreation.employee_demographics;
```



2) selecting distinct members of a specific column

Query:
select distinct gender
from parks_and_recreation.employee_demographics;

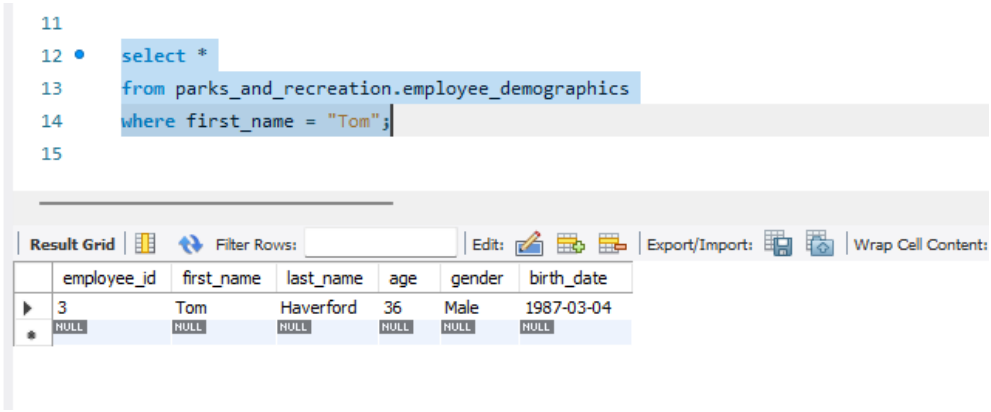


3) Where statement

using where(this is actually used to impose a condition)

Query:

```
select *
from parks_and_recreation.employee_demographics
where first_name = "Tom";
```



Another ex:

```
11
12 •  select *
13    from parks_and_recreation.employee_demographics
14    where (age<45 and gender="Female") or (gender="Male");
15
```

Result Grid | Filter Rows: | Edit: | Export/Import:

employee_id	first_name	last_name	age	gender	birth_date
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
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
NULL	NULL	NULL	NULL	NULL	NULL

4)Like statement

When using '=' we must ensure that the things are fully equal, if for example we want to check "jerry"="jer", the answer is NO. here is where LIKE statement comes in handy

Here , % means the character can be anything, _ means there must be a specific number of characters(if we write “a__” then there is 2 dash(_) ere which indicates we must have 2 slots where any character can sit)
ex:

```
17
18      --   % means the character can be anything,
19      --   _ means the character must be a specific character
20  •   select *
21      from parks_and_recreation.employee_demographics
22      where first_name like "a%";
23
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wra

employee_id	first_name	last_name	age	gender	birth_date
4	April	Ludgate	29	Female	1994-03-27
7	Ann	Perkins	35	Female	1988-12-01
10	Andy	Dwyer	34	Male	1989-03-25
NULL	NULL	NULL	NULL	NULL	NULL

In this example, “a%” means the first character is “a” and then there can be anything after that as we used % here.

Another ex:

```
24
25  •   select *
26      from parks_and_recreation.employee_demographics
27      where first_name like "%a%";
28
```

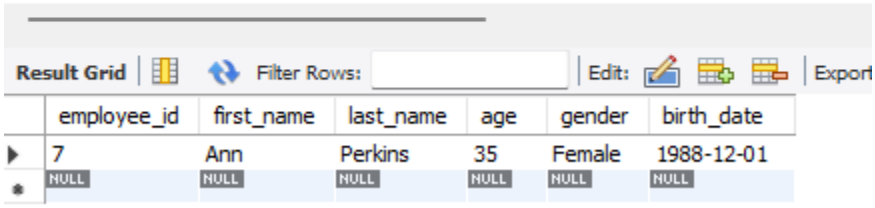
Result Grid | Filter Rows: | Edit: | Export/Import:

employee_id	first_name	last_name	age	gender	birth_date
4	April	Ludgate	29	Female	1994-03-27
6	Donna	Meagle	46	Female	1977-07-30
7	Ann	Perkins	35	Female	1988-12-01
10	Andy	Dwyer	34	Male	1989-03-25
11	Mark	Brendanawicz	40	Male	1983-06-14
12	Craig	Middlebrooks	37	Male	1986-07-27
NULL	NULL	NULL	NULL	NULL	NULL

Anything can be before “a” and anything can be after “a”

Example using “_”:

```
24
25 •  select *
26    from parks_and_recreation.employee_demographics
27    where first_name like "a__";
28
```

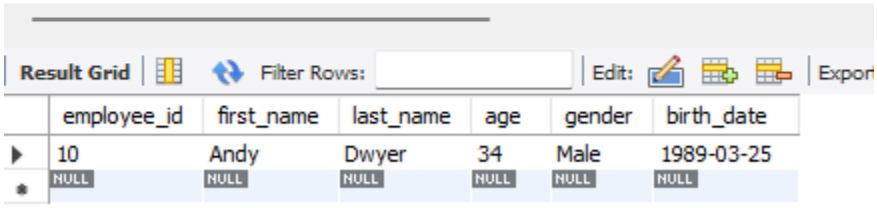


Result Grid | Filter Rows: | Edit: | Export

employee_id	first_name	last_name	age	gender	birth_date
7	Ann	Perkins	35	Female	1988-12-01
NULL	NULL	NULL	NULL	NULL	NULL

After “a” we have 2 dash, so Ann is the only possible name which can be used from the given table

```
24
25 •  select *
26    from parks_and_recreation.employee_demographics
27    where first_name like "a___";
28
```



Result Grid | Filter Rows: | Edit: | Export

employee_id	first_name	last_name	age	gender	birth_date
10	Andy	Dwyer	34	Male	1989-03-25
NULL	NULL	NULL	NULL	NULL	NULL

Here we have 3 _, so Andy is the answer.

Combo of % and _:

```
select *
from parks_and_recreation.employee_demographics
where first_name like "a___%";
```

Result Grid | Filter Rows: | Edit: | Export/Imp

employee_id	first_name	last_name	age	gender	birth_date
4	April	Ludgate	29	Female	1994-03-27
10	Andy	Dwyer	34	Male	1989-03-25
NULL	NULL	NULL	NULL	NULL	NULL

After “a” there are 2 _ and a % which means 2 _ slots must be filled, then you can put whatever you want at last.

5)GroupBy, OrderBy

Groupby is used for grouping rows together that have the same values in the specified columns that we are grouping on(very confusing, use chatgpt to understand and see the examples)

Ex:
Selecting gender from the table. Then grouping the selected column based on gender

Query:
select gender
from parks_and_recreation.employee_demographics
group by gender;

```
-- group by, order by statement
select gender
from parks_and_recreation.employee_demographics
group by gender;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

gender
Female
Male

Ex2:

.***if we are not performing aggregate functions like avg(), min(), max() etc, then the selected columns must match the column name chosen for group by.

```
41      -- group by, order by statement
42 •    select first_name
43      from parks_and_recreation.employee_demographics
44      group by gender;
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
49	02:44:48	select * from parks_and_recreation.employee_de...	3 row(s) returned	0.000 sec / 0.000 sec
50	02:45:50	select * from parks_and_recreation.employee_de...	2 row(s) returned	0.000 sec / 0.000 sec
51	02:47:23	select * from parks_and_recreation.employee_de...	11 row(s) returned	0.016 sec / 0.000 sec
52	21:57:53	select gender from parks_and_recreation.employe...	2 row(s) returned	0.015 sec / 0.000 sec
53	22:00:29	select first_name from parks_and_recreation.empl...	Error Code: 1055. Expression #1 of SELECT list is ...	0.000 sec
54	22:01:47	select first_name from parks_and_recreation.empl...	Error Code: 1055. Expression #1 of SELECT list is ...	0.000 sec

Here we are not using an aggregate function of any kind after the select statement, so first_name can not be chosen because to group by gender we must select gender.

Ex3:

```
41      -- group by, order by statement
42 •    select  avg(age)
43      from parks_and_recreation.employee_demographics
44      group by gender;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Cont

avg(age)
38.5000
41.2857

We used aggregate function here, so the query worked.but the fault is that based on gender we have made group, but gender is not mentioned so we can not understand on what basis group is made.

Solve:

```
41    -- group by, order by statement
42    select  gender,avg(age)
43    from parks_and_recreation.employee_demographics
44    group by gender;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Con

gender	avg(age)
Female	38.5000
Male	41.2857

Ex2:

```
41    -- group by, order by statement
42    select  gender,avg(age), max(age) as maximum_age, min(age) as minimum_age, count(age)
43    from parks_and_recreation.employee_demographics
44    group by gender;
45
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

gender	avg(age)	maximum_age	minimum_age	count(age)
Female	38.5000	46	29	4
Male	41.2857	61	34	7

Ex3:

Query:

```
select occupation, salary
from employee_salary
group by occupation,salary;
```

```
46
47 •  select occupation, salary
48    from employee_salary
49    group by occupation,salary;
```

Result Grid | Filter Rows: | Export: | Wrap Cell

occupation	salary
Deputy Director of Parks and Recreation	75000
Director of Parks and Recreation	70000
Entrepreneur	50000
Assistant to the Director of Parks and Recreation	25000
Office Manager	50000
Office Manager	60000
Nurse	55000
City Manager	90000
State Auditor	70000
Shoe Shiner and Musician	20000
City Planner	57000
Parks Director	65000

Now. Let's look at order by. This sorts the groups we made in ascending order(small to large) by default. To make it descending, we have to use desc.

Ex1:

Query:

```
-- order by sorts things out, it comes after where statement
select first_name, age, gender
from employee_demographics
order by first_name;
```

```
52      -- order by sorts things out, it comes after where
53      select first_name, age, gender
54      from employee_demographics
55      order by first_name;
56
57
```

Result Grid | Filter Rows: | Export: | Wrap Cell Conten

first_name	age	gender
Andy	34	Male
Ann	35	Female
April	29	Female
Ben	38	Male
Chris	43	Male
Craig	37	Male
Donna	46	Female
Jerry	61	Male
Leslie	44	Female
Mark	40	Male
Tom	36	Male

Same example, not order by used on age and we used descending order

```
52      -- order by sorts things out, it comes after where statement
53      select first_name, age, gender
54      from employee_demographics
55      order by age desc;
56
57
```

esult Grid | Filter Rows: | Export: | Wrap Cell Content:

first_name	age	gender
Jerry	61	Male
Donna	46	Female
Leslie	44	Female
Chris	43	Male
Mark	40	Male
Ben	38	Male
Craig	37	Male
Tom	36	Male
Ann	35	Female
Andy	34	Male
April	29	Female

Ex2:
Order by 2 columns at the same time.

Query:
select first_name, age, gender
from employee_demographics
order by gender, age desc;

```
52    -- order by sorts things out, it comes after where statement
53 •  select first_name, age, gender
54    from employee_demographics
55    order by gender, age desc;
56
57
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

first_name	age	gender
Donna	46	Female
Leslie	44	Female
Ann	35	Female
April	29	Female
Jerry	61	Male
Chris	43	Male
Mark	40	Male
Ben	38	Male
Craig	37	Male
Tom	36	Male
Andy	34	Male

6) where vs having

We use ‘having’ after ‘group by’ and ‘where’ before ‘group by’. So if we need to put any conditions over the groups we made using ‘group by’, we must use ‘having’ for that.

Ex:

```
57
58      -- having vs where
59 •    select gender, avg(age)
60      from employee_demographics
61      where avg(age)> 40
62      group by gender;
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
78	20:10:44	select first_name, age, gender from employee_de...	11 row(s) returned	0.000 sec / 0.000 sec
79	20:10:47	select gender, avg(age) from employee_demograp...	Error Code: 1111. Invalid use of group function	0.016 sec

Here we are getting errors as we are using conditions over a group, but 'where' comes before 'group by'. So before making a group we are imposing a condition over that group which makes it error prone. Here comes 'having' to save us.

Query:

```
select gender, avg(age)
from employee_demographics
group by gender
having avg(age)> 40;
```

```
58      -- having vs where
59 •    select gender, avg(age)
60      from employee_demographics
61      group by gender
62      having avg(age)> 40;
```

Result Grid | Filter Rows:

gender	avg(age)
Male	41.2857

Ex2:
Both in one query

Query:

```
select occupation, avg(salary)
from employee_salary
where occupation like "%manager%"
group by occupation
having avg(salary) >50000;
```

```
64
65 •  select occupation, avg(salary)
66    from employee_salary
67    where occupation like "%manager%"
68    group by occupation
69    having avg(salary) >50000
70
71
```

Result Grid | Filter Rows: | Export:

occupation	avg(salary)
Office Manager	55000.0000
City Manager	90000.0000

7) limits and aliasing

Aliasing is used to rename a column when we are using a query.use chatgpt to find out how it works, it' really easy)

Limit helps to select limited number of rows from a column

Ex: selecting first 4 employee with descending age

```
71    -- limits and aliasing
72 •  select *
73    from employee_demographics
74    order by age desc
75    limit 4;
```

Result Grid | Filter Rows: | Edit:

employee_id	first_name	last_name	age	gender	birth_date
5	Jerry	Gergich	61	Male	1962-08-28
6	Donna	Meagle	46	Female	1977-07-30
1	Leslie	Knope	44	Female	1979-09-25
8	Chris	Traeger	43	Male	1980-11-11
NULL	NULL	NULL	NULL	NULL	NULL

Now we want to go to position 2, and then 2 rows after that position.

```
71    -- limits and aliasing
72    select *
73    from employee_demographics
74    order by age desc
75    limit 2,2;
```

Result Grid | Filter Rows: | Edit: | Exp

employee_id	first_name	last_name	age	gender	birth_date
1	Leslie	Knope	44	Female	1979-09-25
8	Chris	Traeger	43	Male	1980-11-11
NULL	NULL	NULL	NULL	NULL	NULL