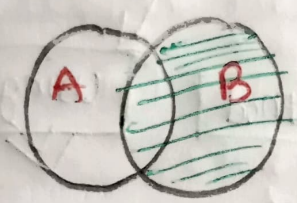
first_name	last_name	money_spent
Boy George	Michael	135.49
David	Bowie	813.17
Blue	Stule	NULL
Bette	Davis	NULL
		450.25

→ 0 0

RIGHT JOIN



selects everything from B along with any matching records from A

① Insert into orders (order_date, amount, ~~customer_id~~)
values (curdate(), 100);

② select * from orders;

id	order_date	amount	customer_id
8	2023-10-01	100.00	NULL

Ex. → Select first_name, last_name, order_date, amount
FROM customers RIGHT JOIN orders on customers.id
= orders.customer_id;

first_name	last_name	order_date	amount
. .	. .	. .	. .
Bette	Davis	1999-04-11	450.25
NULL	NULL	2023-10-11	100.00

ON DELETE CASCADE

5. How can you delete a customer name say 'George' and its corresponding order details. As both are linked by Foreign key?

Ans → Delete From customers where last_name = 'George';

✗ Fails

output → cannot delete or update a row a foreign key constrain fails

Two methods we have

↙ using update method

↘ using Delete cascade

lets recreate the tables;
First we need to delete orders, customers tables

then

create table customers(same synatax)

In orders table

```
CREATE TABLE ORDERS ( id INT PRIMARY KEY AUTO_
                      INCREMENT,
order_date DATE, amount DECIMAL (8,2),
customer_id INT, FOREIGN KEY (customer_id)
REFERENCES customers(id)
ON DELETE CASCADE
);
```

If a corresponding usn-name is deleted its corresponding order will also be deleted; (Hooray)

Now,

1) delete from customers where last_name = 'George';

✓

2) select * from customers

id	first_name	last_name	email
1	Boy	George	. .
2	. .	. .	. .
3			
4			
5			

Succefully deleted

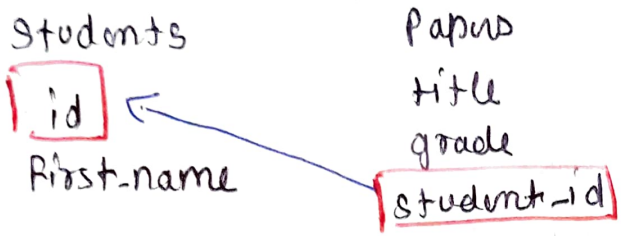
3) select * from orders;

id	order_date	amount	customer_id
3	2014-12-12	800.67	2
4	2015-01-03	12.50	2
5	1999-04-11	450.25	5

corresponding order details also deleted with id that = id

Exercise

① write the schema ?



ans ⇒ create table students (id INT PRIMARY KEY AUTO_INCREMENT,
First_name VARCHAR(50));

CREATE table papers (title VARCHAR(50), grade INT,
student_id INT, FOREIGN KEY (student_id) REFERENCES
students (id));

Q.2

First_name	title	grade
Samantha	DE . . .	98
Samantha	Russian	94
Charlos	Borges	89
Caleb	My second	75
Caleb	My First	60

ans ⇒ select first_name, title, grade
from Students . ;

Must practise Exercise -13

MANY TO MANY

first_name	average	passing_status
Samantha	96.000	PASSING
Carlos	89.000	PASSING
Caleb	67.500	PASSING
Raj	0	FAILING
Lisa	0	FAILING

```
Select first_name, IFNULL( AVG(grade), 0) as average,
CASE
when IFNULL (AVG(grade), 0) >= 60 then 'PASSING'
else 'FAILING'
end as passing_students
From students. id = papers. student_id
GROUP BY first_name
Order by average desc ;
```

Section-14

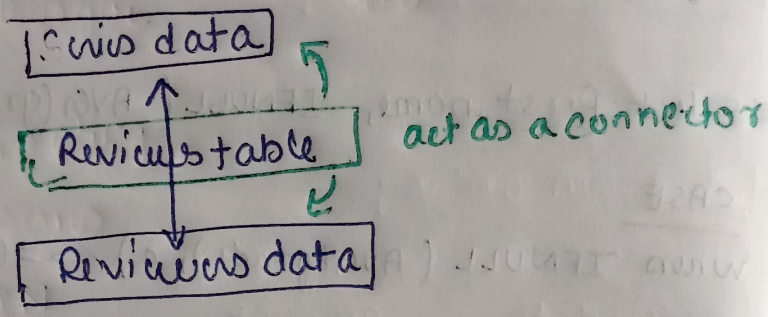
MANY TO MANY

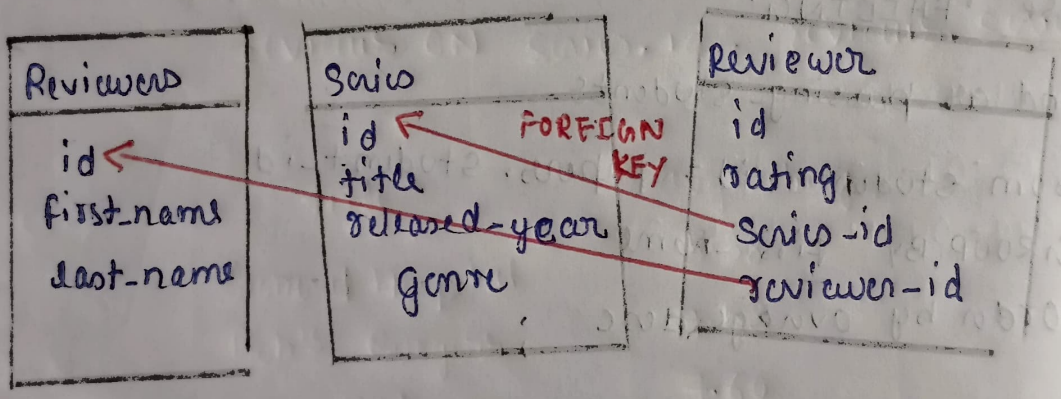
Books ⇌ Authors

Blog post ⇌ Tags

Students ↔ classes

6. Imaging we are building a t.v show reviewing application.





Reviewers

id	first_name	last_name
1	Blue	Stule
2	Wyatt	Earp

Series

id	title	released_year	genre
1	Archer	2009	Animation
2	Fargo	2014	Drama

Reviews

id	rating	reviewer-id	movie-id
1	8.9	1	2
2	9.5	2	2

5. Create New database movies_db and then three tables Reviews, Series, Reviews linked together by foreign key.

① CREATE TABLE reviewers(id INT PRIMARY KEY AUTO_INCREMENT,
First_name VARCHAR(50) NOT NULL,
last_name VARCHAR(50) NOT NULL,);

② CREATE TABLE series (id INT PRIMARY KEY AUTO_INCREMENT,
title VARCHAR(50), released_year YEAR, genre VARCHAR
(100));

③ CREATE TABLE reviews(id INT PRIMARY KEY
AUTO_INCREMENT,
rating DECIMAL (2,1),
series_id INT,
reviewer_id INT,
FOREIGN KEY (series_id) REFERENCES series(id),
FOREIGN KEY (reviewers_id) REFERENCES reviewers(id)
);

Q.1

title	rating
Archer	8.0
Archer	7.5
Archer	8.5
Archer	7.7
Archer	8.9
Arrested Development	8.1
"	6.0
"	8.0
Bob's Burger	7.0

code → Select title, rating
From Series
JOIN
reviews ON series.id = reviews.series_id;

Q.2

title	avg_rating
Criminal Hospital	5.38
Bob's Burgers	7.58
Seinfeld	7.60
Bojack Horseman	7.94
Arrested Development	8.08
Curb your Enthusiasm	8.12

code →
Select
title, ROUND (AVG (rating), 2) as
avg_rating
FROM

```
series
    JOIN
reviews ON  series.id = reviews.series_id
GROUP BY title
ORDER BY avg-rating ;
```

first_name	last_name	rating
Thomas	Stoneman	8.0
Thomas	Stoneman	8.1
Thomas	Stoneman	7.0
Thomas	Stoneman	7.5
Thomas	stoneman	9.5
wyatt	Skaggs	7.5
wyatt	skaggs	7.6
wyatt	Skaggs	9.3

practise
From
PPT

S.Q.L

PART-2

Section-15

Introduction views

Q. what is a review ?

Ex. → lets create a table

ramu

```
Select title, released_year, genre, first_name, last_name FROM
reviews JOIN series ON series.id = reviews.series_id
JOIN reviewers ON reviewers.id = reviews.reviewer_id;
```

title	released_year	genre	rating	first_name	last_name
Archer	2009	Animation	8.0	Thomas	Storeman
Arrested development	2003	Comedy	8.1	"	"
Bob's burger	2011	Animation	7.0	"	"
Bojack Horseman	2014	Animation	7.5	"	"
Breaking Bad	2008	Drama	9.5	"	"
Archer	2009	Animation	7.5	wyatt	Skaggs
Bojack Horseman	2014	Animation	7.6	wyatt	"
Breaking Bad	2008	Drama	9.3	"	"
Curb your enthusiasm	2000	Comedy	6.5	"	"
"	2000	Comedy	8.4	"	"
Fargo					
. .					
. .					
. .					

→ the view command will create a virtual review of this table;

```
CREATE VIEW full_reviews AS
```

ramu

Show tables; → full_reviews table will be created;

Now we can do opration on this New table, like group by, select, order by etc...

* (atleast when it comes to selecting)

we cannot to all sql oprations on this table

Ex→ DELETE FROM full_reviews WHERE released_year = 2010;

Error 1395 (HY000) cannot delete from join view 'tv_db.full_reviews'

A smaller portion of views are updatable and treatable

Any Functions in views like SUM(), MIN(), MAX(), COUNT(), DISTINCT, GROUP BY, HAVING, UNION OR UNION ALL etc...

Ex. 2→ Select * From Series

There are 14 ~~columns~~ rows

lets create this as a view

```
CREATE VIEW ordred_series as
Select * from series;
```

lets insert→ Insert into ordred_series (title, released_year, genre)

Values ('The Great', 2020, 'comedy');

✓ WORKS

It works as it didn't break any rule

DELETE FROM ordred_series WHERE title = 'The Great';

✓ WORKS

Replacing / Altering views

Q. what if want to arrange the views in desc order ?

(i) create view ordered_swis as

Select * FROM Swis ORDER BY released_year DESC;

Output → Error table already exists

What we do ?

CREATE OR REPLACE VIEW ordered_swis as

Select * From swis order by released_year DESC;

✓ Now it will work

or

ALTER VIEW ordered_swis as Select * From Swis order by released_year;

Both ALTER / REPLACE can be used to give the same result.

→ DROP VIEWS ordered_swis;
to drop a view

GROUP BY HAVING

→ used to filter groups that we get back from group by

Q. Select title, AVG(rating) FROM full_reviews GROUP BY title;

title	AVG(rating)
Archer	8.1200
Arrested Development	8.0800
Bob's Burger	7.5200
..	..
..	..

12 rows

Q. What if we want to limit the table to a particular row
For example only the movies having more than 3 user rating
is this possible ? (From a temporary table 🐭)

✗ WHERE clause Not work here

✓ yes! Having will be used

Code → Select title
AVG(rating),
COUNT(rating) AS review_count
FROM full_reviews
GROUP BY Title HAVING COUNT(rating) > 1 ;

title	review_count	avg(rating)
Archer	5	8.12
Arrested Develop	5	8.08
Bob's Burger	5	7.52
..	..	..

GROUP BY ROLLUP

with ~~roll up~~ rollup only works with group by

Select avg(rating) From full reviews ;

avg(rating)
8.02553

} सबका average लेके direct one value मिलेगा।

Select avg(rating), title from full_reviews group by title ;

Command ⇒ select avg (rating), title from reviews
group by title;

Avg(rating)	title
8.1200	Archer
8.0800	Arrested Development
7.5200	Bob's Burgers
9.3600	Breaking Bad
8.12	Cobb . . .

(1)

Individually average point करेगा।

Command ⇒ Select title, avg(rating) FROM Full_reviews
GROUP BY title WITH ROLLUP ;

Output table (1)

last में सबका avg. दे देता है

NULL	8.02553

Summary static/ avg. For the whole table →

V.V.I

Group by Group by के साथ जो चिज है उसको select के बाद रखना है।

Command ⇒ select count (rating) . . .

count (rating)	title
5	Archer
..	..
..	..
47	NULL

Select released-year, genre, Avg(rating) FROM full-reviews

this GROUP BY released-year, genre with ROLLUP;
is avg. rating for each released-year & genre combination but also for each released year and all rows

(5)

released-year	genre	Avg(rating)	
1963	DRAMA	5.38	→ avg. rating for all g. drama in 1963
1963	NULL	5.38	→ avg. rating for all genre in 1963.
1989	COMEDY	7.60	
1989	NULL	7.60	
..	..	..	
NULL	NULL	8.02553	

avg. for all of the ratings

Select first_name, released-year, genre, avg(rating) from full-reviews group by released-year, genre, first_name with rollup;

first-name	released-year	genre	avg(rating)
colt	1963	Drama	4.50
Domingo	1963	Drama	5.80
kimbra	1963	Drama	6.80
Pinkie	1963	Drama	4.30
ayatt	1963	Drama	5.50
NULL	1963	Drama	5.50
NULL	1963	NULL	5.38
Domingo	1989	Comedy	7.20
..	..	..	..
NULL	NULL	NULL	8.02553

SQL MODES

These are the settings that we can turn on/off to change the behaviour and validation of MySQL.

Q. How many scopes do we have for a sql mode ?

two

No spacing viewing modes

(1) Select @@ GLOBAL . sql_mode ; ← permanent

(2) Select @@ SESSION . sql_mode ; ← temporary

Q. How to set the modes ?

(1) SET GLOBAL sql_mode = 'modes' ;
SET SESSION sql_mode = 'modes' ;

ONLY_FULL_GROUP_BY, STRICT_TRANS_TABLES, NO_ZERO_IN_DATE,
ERROR_FOR_DIVISION_BY_ZERO, NO_ENGINE_SUBSTITUTION

(1) SELECT @@ SESSION . sql_mode ;

Select 3/0 ;

3/0
NULL

1 row in set, 1 warning

Show warnings

Level	code	Message
Warning	1365	Division by 0

Q How to disable all the modes?

SET SESSION sql_mode = ' ';

Q How if i want to keep all modes except error for divison by 0 ?

SET SESSION sql_mode = 'ONLY_FULL_GROUP_BY, ...

NO_ENGINE_SUBSTITON' ;

SELECT All except ERROR_FOR_DIVISON_BY_ZERO

→ Query OK, 0 rows affected

Now Select 3/0;

3/0
NULL

1 row in set  no warnings Now

Show warnings;

Empty set

STRICT TRANS TABLE

DESC Reviews;

Field	Type	Null	key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
rating	decimal(2,1)	yes		NULL	
Series_id	int	yes	MUL	NULL	
reviewer_id	int	yes	MUL	NULL	

Q. What if insert into reviews(rating) values ('Hi');

Output → Error 1366, Incorrect decimal value for 'Hi' column 'rating' at row 1

Q. what if STRICT_TRANS_TABLE mode is disabled ?

① set session sql-mode = '';

Select @@ SESSION.sql-mode;

@@ SESSION.sql-mode

↑ empty set

Insert into reviews(rating) values ('hi');

→ Query OK, 1 row affected, 1 warning

Select * From reviews;

48	0.0 ↑ printed	NULL	NULL

why our values auto transformed into a decimal value ≈

SOME MORE SQL Modes

① ONLY_FULL_GROUP_BY

Watch the tutorial ?
by colt steel

Section-16

Introducing windows functions

Lets take a review of group by

```
Select department, Avg(salary) FROM emps group by department;
```

emp_no	department	salary
8	Sales	59,000
12	Sales	60,000
20	Customer Service	56,000
21	customer Service	55,000

↓

Sales group

8	Sales	59,000
12	Sales	60,000

↓

Customer Service group

20	Customer Service	56,000
21	customer Service	55,000

↓

department	Avg(salary)
Sales	59,500.00
customer service	55500,00

group by collapses each group of rows into a single result row

WINDOW Function → window function performs aggregate operations on group of rows, but they produce a result FOR each row.

```
Select emp_no, department, salary, Avg(salary)
OVER (PARTITION BY department) AS dept_avg FROM emps;
```

emp_no	department	salary
8	Sales	59,000
12	Sales	60,000
20	Customer Service	56,000
21	customer service	55,000

Sales window

8	Sales	59,000	dept avg 59,500
12	Sales	60,000	

Customer Service window

20	Customer Servic	56,000	dept_avg 55,500
21	Customer Servic	56,000	

emp-no	department	Salary	dept_avg
20	customer servic	56,000	55,500
21	"	55,000	55,500
8	Sales	59,000	59,500
12	Sales	60,000	59,500

→ Result for each row

OVER

The OVER() clause constructs a window. when its emptied the window will ~~record~~ include all records.

→ Select * From employees;

emp-no	department	Salary
1	engineering	80,000
2	"	69,000
3	"	70,000
4	"	103000
5	"	87000
6	"	89000
7		91000
8	sales	59,000
9	"	
10	"	
11	"	
12	"	
13	"	
14		
15	customer servic	38,000
upto 21	"	

Select Avg(Salary) OVER() From employees;

Avg(Salary) Over()
68428.5714
"
68428.5714

21 rows (total)

सारे employees के corresponding उसका avg. show होगा।

① Select emp-no, department, salary, avg(salary) OVER() FROM employees;

↑ only with aggregate functions

emp-no	department	Salary	Avg(salary) OVER()
1	enginuring	80,000	68,428.5714
2	"	69,000	68,428.5714
3	"	70,000	68,428.5714
4	"	1,03,000	"
5	"	67,000	"

Ex. 2 ⇒ " MIN(salary) OVER(), Max(salary) OVER() From employees;

emp-no	department	Salary	MIN(salary) OVER()	Max(salary) OVER()
1	engineering	80,000	31,000	159000
2	enginuring	69,000	31,000	159000

When we not called min/max as windowed function

Select MIN(salary), Max(salary) FROM employees;

MIN(salary)	Max(salary)
31,000	1,59,000

PARTITION BY

Q. what if we only want the department 'engineering' through OVER() method ?

Syntax

->

AVG(salary) OVER (PARTITION BY department)

* Inside of the OVER() , use PARTITION BY to form rows into groups of row

command -> select emp-no, department, salary, AVG(salary) OVER(PARTITION BY department) AS dept-avg FROM employees;

emp-no	department	Salary	dept_avg
15	customer service	38,000	46,571.4286
16	customer service	45,000	46,571.4286
17	customer service	61,000	46,571.4286
18	customer service	40,000	"
19	customer service	31,000	"
20	customer service	56,000	"
21	customer service	55,000	"
1	engineering	80,000	81,285.7143
2	"	69,000	81,285.7143
3	"	"	"
4	"	"	"
5	"	"	"

OR

Select department, AVG(salary) FROM employees GROUP BY department;

department	AVG(salary)
engineering	81,285.7143
Sales	77,428.5714
Customer service	46,571.4286

```
Select emp_no, department, salary,
SUM(salary) OVER(PARTITION BY department) AS dept_payroll,
SUM(salary) OVER () AS total_payroll
FROM employees;
```

emp_no	department	salary	dept_payroll	total_payroll
15	customer service	38,000	326000	1437000
16	"	45,000	326000	1437000
17	"	61,000	326000	1437000
18	"	40,000	326000	"
19	"	31,000	326000	"
20	"	56,000	326000	"
21	"	55,000	326000	"

ORDER BY WITH WINDOWS

OVER(ORDER BY salary DESC)

Use ORDER by inside of the over () clause to re-order rows within each window.

Example ⇒

```
Select emp_no, department, salary
SUM(salary) OVER(PARTITION BY department order by
order by salary desc) AS rolling_dept_salary
FROM employees;
SUM(salary) OVER(PARTITION BY DEPARTMENT) as
total_dept_salary
FROM employees
```

emp_no	department	salary	rolling_dept_salary	total_dept_salary
17	customer service	61,000	61,000	326000
20	"	56,000	117000	326000
21	"	55,000	172000	326000
16	"	45,000	217000	"
18	"	40,000	257000	"
15	"	38,000	295000	"

4	engineering	103000	103000	569000
7	engineering	91000	194000	"
6	engineering	89000	283000	"
1	engineering	80000	363000	"
3	engineering	70000	433000	"
2	Engineering	69000	502000	"

Ex. 2

```
Select emp-no, department, Salary
MIN(Salary) OVER(PARTITION BY department ORDER BY
Salary DESC) as rolling_min
FROM employees;
```

emp-no	department	Salary	rolling-min
17	customer service	61,000	61,000
20	"	56,000	56,000
21	"	55,000	55,000
4	Engineering	103000	103000
7	"	91000	91000
6	"	89000	89000
1	"	80000	80000

RANK()

```
Select emp-no, department
Salary,
RANK() OVER( ORDER BY Salary DESC) as overall_
Salary_rank FROM employees;
```

→ Rank is only going to work

emp-no	department	salary	Overall-salary-rank
10	Sales	159000	1
4	Engineering	103000	2
7	"	91,000	3
6	"	89,000	4
11	"	80,000	5
3	Sales	72,000	6
9	Sales	70,000	7
2	Engineering	69,000	8
5	Engineering	67,000	9
..	..	..	..

```
Sulct emp-no, department, salary                    as dept_salary-rank
RANK()  OVER( PARTITION BY department ORDER BY Salary)
RANK()  OVER( ORDER BY Salary desc) as Overall-
                Salary-rank

FROM employee;
```

emp-no	department	Salary	dept_salary-rank	Overall-salary-rank
10	Sales	159000	1	1
4	Engineering	103000	1	2
7	"	91,000	2	3
6	"	89,000	3	4
1	"	80,000	4	5
11	Sales	72,000	2	6
3	engineering	70,000	5	7
9	Sales	69,000	3	7
2	engineering	67,000	6	9
5	Engineering	61,000	7	10
19	Customer service	31,000	1	11
..	..	..	..	..

Select
emp-no, department, salary RANK () OVER (...),
RANK() OVER (...) FROM employee ORDER BY
department;

emp-no	department	Salary	dept-Salary-rank	overall-salary-rank
17	customer servic	81,000	1	11
20	"	56,000	2	16
21	"	55,000	3	17
..	..	..	..	18
				..
4	engineering	103000	1	2
7	"	91,000	2	3
6	"	"	3	4
..	..			
10	Sales	159000	1	1
11	Sales	72000	2	6
9	Sales	70,000	3	7

ROW_NUMBER

Returns the number of the current row within its partition

DENSE RANK

This rank doesn't skip any rank if there's a type they resume from the next number.

```
Select
    emp_no,
    department,
    Salary,
    Row Number () OVER (PARTITION BY department ORDER BY
    SALARY desc) as dept_row_number,
    RANK() OVER(PARTITION BY department ORDER BY SALARY
        DESC) as dept_Salary_rank
```

```
RANK() OVER( ORDER BY salary desc) as ovnall_rank ,
DENSE_RANK() OVER(ORDER BY salary desc) as ovnall_dense_rank,
ROW_NUMBER() OVER( ORDER BY salary desc) as ovnall_num
FROM employees ORDER BY ovnall_rank;
```

ovnall_num

emp_no	department	Salary	dept_row_number	dept_salary_rank	ovnall_rank	Ovnall_dense_rank
10	Sales	159000	1	1	1	1
4	enginering	103000	1	1	2	2
7	"	91000	2	2	3	3
6	"	89000	3	3	4	4
1	"	80000	4	4	5	5
11	Sales	" 70,000	2	2	6	6
3	enginering		5	5	7	7
9	Sales		3	3	7	8
2	enginering		6	6	9	9
5	enginering		7	7	10	10
17	customer service	61,000	1	1	11	11
13	Sales	61,000	4	4	11	12
14	Sales	60,000	5	4	11	13
12	Sales		6	6	14	14
8	Sales		7	7	15	15
20	customer service		2	2	16	16
21	"		3	3	17	17
16	"		4	4	18	18
18	"		5	5	19	19
15	"		6	6	20	20
19	"		7	7	21	21

NTILE

Divides a partition into N groups (buckets).

If, for example N is 4 NTILE, divides rows into four buckets.

```
Select
 emp_no, department, salary
 NTILE(4) OVER( PARTITION BY department ORDER BY salary
DESC) as dept_salary_quartile,
NTILE(4) OVER( ORDER BY salary DESC) AS salary_quartile
 FROM employees;
```

emp_no	department	Salary	dept_salary_quartile	salary_quartile
10	Sales	159000	1	1
4	engineering	103000	1	1
7	engineering	91000	1	1
6	engineering	89000	2	1
1	engineering	80000	2	1
11	Sales	72000	1	1
3	engineering	70000	3	2
9	Sales	70000	2	2
2	engineering	69000	3	2
5	Engineering	67000	4	2
17	Customer Service	61000	1	2
13	Sales	61000	1	2
14	Sales	61000	2	3
12	Sales	60000	3	3
8	Sales	59000	3	3

FIRST_VALUE

```
Select emp_no, department, salary
  FIRST_VALUE (emp_no) OVER (PARTITION BY department ORDER
  by salary DESC) as highest_paid_dept,
  FIRST_VALUE (emp_no) OVER (ORDER BY salary DESC) as highest
  _paid_overall
     FROM employees;
```

emp_no	department	salary	highest_paid_dept	highest_paid_overall
10	Sales	159000	10	10
4	engineering	103000	4	10
7	"	91000	4	10
6	"	89000	4	"
1	"	80000	4	
11	Sales	72000	10	
3	engineering	70000	4	
9	Sales	70000	10	"
2	engineering	69000	4	
5	"	67000	4	
17	customer service	61000	17	
13	Sales	61000	10	"
14	"	61000	10	
12	"	61000	10	"
8	"	61000	10	

LEAD / LAG

For comparison b/w next row

(i) Select emp_no, department, salary,
LAG(salary) OVER (ORDER BY salary DESC)
FROM employees;

emp_no	department	Salary	LAG(salary) OVER(ORDER By Salary DESC)
10	Sales	159000	NULL
4	engineering	103000	159000 ← Previous value
7	"	91000	103000
6	"	89000	91000
1	"	80000	89000

we can calculate the difference

(ii) Select "
Salary - LAG(salary) over (") as salary_diff
From employees;

emp_no	department	Salary	Salary diff
10	Sales	159000 (2)	NULL (1-2)
4	engineering	103000 (1)	-56000
7	"	91000	-12000
6	"	89000	-2000
1	"	80000	-9000
11	Sales	72000	-8000

(iii) Select "
Salary - LEAD(salary) .. FROM employees;

emp-no	department	salary	salary - diff
10	Sales	159000 (1)	56000 (1-2)
4	engineering	103000 (2)	12000
7	"	91000	2000
6	"	89000	9000
1	"	72000	8000
11	Sales	70000	2000

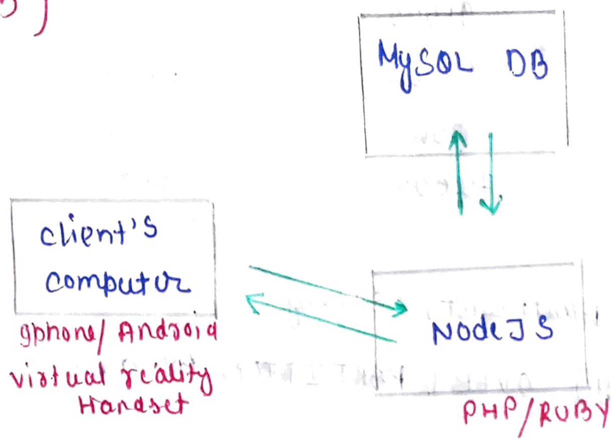
(iv) Select emp_no, department, salary,
Salary - LAG(salary) OVER(PARTITION BY department ORDER BY salary DESC) as dept- salary_diff FROM employees;

emp-no	department	salary	dept- salary - diff
17	Customer service	61,000	NULL
20	"	56,000	-5000
21	"	55,000	-1000
16	"	45,000	-10,000
4	engineering	103000	NULL
7	"	91000	-12,000
6	"	89000	-2000

Section-17 | Section-18 | 9g data_clone

Practise → Mysql and NodeJS

(Section-35)



Q. what is node ?

node is Javascript it uses exact same syntax but it can be used on backend for

✓ mysql-ctl start

✓ mysql-ctl stop

* mysql-ctl cli ✓

Skip the password

to run a sql file in terminal → Source filiname.sql

groom.io

Username→ Shivam 181813 2@gmail.c

ls/ cd

Q. to print Hello.world ?

(1) console.log ('Hello world') ;
↳ to print Hello world

(2) For (var i=0 ; i<500; i++)
{ console.log ('Hello world') ; }

(3) console.log (5+5);

output → 10

Root user : root

Database name : mysql