DBM-04 28/02/2025

URBAN EDGE 28/02/2025 FRIDAY

★ SELECT Syntax Errors:

★ Order of Execution:

F	J	W	G	H	S	D	O	L
From	Join	Where	GroupBy	Having	Select	Distinct	OrderBy	Limit

DDL (Exercises & Assignments)

★ LIKE Thy-out

Imp | DATE comparison.

URBAN EDGE

Query: Display Id, EName and DOJ of the Employee(s) who joined in the year '2014'.

Q1 USING LIKE (LIKE)

```
> SELECT Id, EName, DOJ FROM Employee
  WHERE DOJ LIKE '2014-__-__';
```

Equivalent both.

[Exact math of month, year, date ke liye LIKE use ho skta.

But many times.

Q2 Relational operators with DATE.

```
> SELECT Id, EName, DOJ FROM Employee
  WHERE DOJ > '2013-12-31'
  AND
        DOJ < '2015-01-01';
```

★ Functions

- Numeric
- Character
- Conversion
- → Date functions
- → Aggregate functions
- → Miscellaneous.

Single Row Fn's	Multi-Row Fn's
Return operates on used in — single row / single ; select, where, group by, order by, having	single row / Multiple rows ; Select, Group by, Having

1. NUMERIC (Single Row function) All

CEIL — FLOOR — ABS — ROUND

ROUND — Round (num) — round to nearest integer
Round (num, 1) — Rounds to 1 decimal place
ROUND (num, 2) — Rounds to 2 decimal places.

★ CEIL and FLOOR of -ve Nos dekh leno ek baar.

FLOOR → nearest Lower Integer value

CEIL → nearest upper integer value.

ABS → Distance from 0

can take int, flot, double anything and returns also same but +ve

★1. Exercise_functions_ceilingFloor

★ first Question Doubt
1) Tables toh dekhne hi padenge
2) Yaha bola for every vendorld, Productld unique combination
(Dhyaan se dekh
enka matlab hai
har vendorld ke liye
kyuki har vendorld ke
sath unique Product_ld
hi toh combined logic).

2. Ex Absolute

2. Character Functions (

- UPPER (value)
  ↑ column or single value
- LOWER (value)
  ↑ column or single value
- LENGTH (value) → It counts blank spaces as well.
  ↑ column or single value
- CONCAT (str1, str2) or ★ Imp || (pipe operator)
  ↑ col1 ↑ col2
  ↓ Sequence1 || Sequence2 || Sequence3

• SUBSTR (value , start_position, end_position) ★ Don't mix it with zero index

column_name or single value ↑ if not then by (default) end of string.

Yaha bas position hai 1 se count karta ja.

ex. SUBSTR ('DATABASE', 5) → 'BASE'

SUBSTR ('DATABASE', 3, 3) → 'TAB'

SUBSTR (city, 1, 4); → (LOND)

column_name

LONDON
1234

★ IMP - if the first Arg > Length → NULL

★ To get 10 characters from the 2nd position.

SUBSTR (city, 2, 10);

To Be

ex:- TO_CHAR (MinTemp)

★ Jyada usage DATE ke liye hai

★ But it can convert other datatypes also to CHAR simply

3● Conversion Functions:

•1. TO_CHAR → to convert data from one format to other.

date (usually)

DATE: i) To convert the date data type attribute to other format.

City	Country	MinTemp	RecordDate
London	UK	-10.00	1990-01-25

> SELECT TO_CHAR (RecordDate, 'MM/DD/YYYY') "AMERICAN_FORMAT" FROM Weather;

• TO_DATE

Output: AMERICAN_FORMAT
01/25/1990
01/20/1991

★ Any DATE data type attribute from MSQL format can be converted to user-defined format using TO_CHAR conversion function.

#2 Parts of Date

↳ To fetch the parts of the DATE like date, month, year.

URBAN EDGE

```
> SELECT TO_CHAR (recordDate, 'DD') "DATE",
       TO_CHAR (recordDate, 'MM') "MONTH",
       TO_CHAR (recordDate, 'YY') "YEAR"
  FROM Weather;
```

DATE	MONTH	YEAR
25	01	90
-	-	-

#3 Parts of DATE in desired format:

```
> SELECT TO_CHAR (recorddate, 'DAY')   "DAY"
       . TO_CHAR (recorddate, 'Month') "Month"
       TO_CHAR (recorddate, 'Mon')     "Mon"
```

DAY	MONTH	MON	YEAR
Monday	January	Jan	1990
Sunday	January	Jan	1991

#2 TO_DATE:

↳ To convert user-defind format of date-time to MySQL format. (You have to specify the format in which your Date is then only it can convert it by fetching DD, YYYY and MM from it.)

String | VARCHAR | CHAR

CAST (bhi use kar sakte hai)

TO_DATE (first Arg, 2nd Arg) (MENDATORY) two args.

first Arg: your format date

2nd Arg: give format of your Date.

```
TO_DATE (RecordDate, 'DD/MM/YYYY')
```

as it is CHAR now.

* Now DATE functions can no longer be used for this

SUMMARY of TO_CHAR:

-> It can be used to convert any data type possible to CHAR.

* Specifically

-> It is used to convert DATE data type to user-defined format.

-> It can be used to fetch parts of DATE in desired format.

* AGGREGATE FUNCTIONS : * (Operate on Multiple rows and Return a single Row)

SUM MAX MIN COUNT AVG

URBAN EDGE

COUNT(city)
↓
no. of rows in that column
(→ includes duplicates)
(→ ignores NULL)

COUNT(*)
↓
No of rows in Entire table
(→ Includes duplicates)
(→ Includes NULL as well)

Imp

* [All Aggregate functions Ignore NULL values except COUNT(*)]

* SUM (total), AVG (total) only on NUMERIC datatypes

* Go Try out

* MIN, MAX, COUNT on all datatypes.

* To count only @ DISTINCT rows in that column.

COUNT(DISTINCT (CITY))

* AVG → next page [P.T.O]

* DATE FUNCTIONS

↳ 1> MONTHS_BETWEEN (date1, date2);

↓ gives decimal result → if date1 < date2 ⇓ -ve

date1 - date2

Must be Date Type not Strng / char / varchar Type or cannot even be Strng literals of Date format

* AVG (ignores NULL)

URBAN EDGE

> SELECT AVG(MinTemp) "AVERAGE",
SUM(MinTemp) / COUNT(*) "Avg_over_All_Rows"
SUM(MinTemp) / COUNT(MinTemp) "Avg_Excludg_NULLs";

Includes NULLs only COUNT(*)

∴ AVG(MinTemp) ≡ SUM(MinTemp) / COUNT(MinTemp).

5# DATE functions:

→ Adding a duration to data

→ Finding time difference b/w two dates

1) SYSDATE → Returns CURRENT timestamp (date and time of the System)

Not a function but a column

> SELECT SYSDATE FROM Information_Schema.System_Users;

2) CURRENT_DATE (Returns current date of the system)

// same

3) Add_Months:

> SELECT ADD_MONTHS(RecordDATE, 6) "After_6_months",
ADD_MONTHS(RecordDate, -12) "Befor_12_months";

4) MONTHS_BETWEEN; → Double value returned
→ if date1 < date2
MONTHS_BETWEEN(date1, date2); -ve.

4) Parts of date in desired format.

> SELECT DAY (recorddate),
MONTH (recorddate),
YEAR (recorddate) 'Year' from test

//

RS | 1990

ORDER BY : SYNTAX : ORDER BY <columnname>,

~~Note~~ things :

1) Order By → default order → ASC
2) ORDER BY → Single column

3) ORDER BY → multiple columns with same sorting order

↓

> ORDER BY DEPT, Designation;

★ SORTED first on DEPT
then
wherever clash SORTED based on Designation (But Both in ASC → default)

4) ORDER BY → multiple columns with different sorting order :

> ORDER BY DEPT ASC, Designation DESC;

★ ORDER BY → POSITIONAL string :

Order by on the basis of column in 2nd position of SELECT Query

> SELECT Id, EName FROM Employee
ORDER BY 2;

↓ will be sorted based on EName.

Also can be like this :

> ORDER BY 2, 3;
two columns

Go See Example and Try out

Output city

R1-R2
-2.097
-12.03
-11.1
-2
-11.06

X 6 ORDER BY based on Column that is not there in SELECT clause

URBAN EDGE

> SELECT Id, FnName FROM Employee ORDER BY SALARY;

✓ it works

ORDER OF EXECUTION

F J W G H S D O L

FROM JOIN WHERE Group By HAVING SELECT DISTINCT Order By Limit.

#

Quiz - functions, Quiz - Order By

Quiz do ek question hai pyaara

#8. GROUP BY — HAVING

Need for Aggregate functions and WHERE can do stuff like this:
(min, max, avg, sum, count)

Group By and Having

Q: Find the total salary paid in ICP department?
> SELECT SUM(salary) "Total_Salary" FROM Employee
WHERE Dept = 'ICP';

What if we want to find the total Salary in each dept.
↳ GROUP BY → to achieve such results in Single Query

"Helps to Group data, and apply aggregate function on each group."

SYNTAX:

```
SELECT << Grouping Columns and Aggregate functions >>
FROM <table_name>
GROUP BY << Grouping_criteria >>
```

* One Row for each group is generated.
* Ordering is not fixed ——> Just use ORDER BY for that.

Q: To get the Total Salary of Each Dept
> SELECT Dept, SUM(salary) FROM Employee GROUP BY Dept;

GROUP BY on more than 1 Column:

Required: No. of Employees working in a department under each Manager

* COUNT(*) kyuki count ~~employees~~ kar rahe empId se ~~So ke NULL ya Duplicate~~ NA

1> SELECT Dept, COUNT(*) FROM Employee GROUP BY Dept;
X because this gives the no. of employees in each dept
2> SELECT Manager, COUNT(*) FROM Employee GROUP BY Manager;
X because this gives no. of Employees under each Manager.

*
3> SELECT Dept, Manager, COUNT(*) FROM Employee
GROUP BY Dept, Manager; ✓

★ NULL is also grouped By

URBAN EDGE

Important :

All previous Problems done, there was only one col, now returned.

1) You can't use/display any other Column while using Aggregate function in a SELECT. (while not using GROUP BY)

Doubt : If you say

```
SELECT EmpName, COUNT(EmpId)
FROM Employee
WHERE Dept = 'ICP';
```

1) from employee table
2) All rows selected when Dept = ICP
3) Now COUNT karke -> 1 value come.

CustName
Avinash
Nemi
Vinni

Single Value

So error

Not Possible,

REASON

Aggregate functions
Operates -> Multiple Row
Return -> Single Row

Simf

2) Aggregate Functions WHERE naal ni chalde hai

when we want only Single Row, Single Column Result

functions -> me both examples.

3) More than (1) columns bhi aa skte hai. Simf Aggregate and WHERE use kar ke (But dono chije select me Aggregates hi honge) (no bhi non-related)

*2. GROUP BY Example

*1 Aggregate functions without Group By

URBAN EDGE

* If only Aggregate fields in the SELECT clause

↓

the GROUP BY is NOT necessary.

* Depending on Requirement GROUP BY can be written or skipped in such case

*2 Q: To display Dept and no. of employees working in coll dept:

> SELECT Dept, count(id) AS "No of Employees"
GROUP BY Dept;

✡ When Non-Aggregates + Aggregates in SELECT then GROUP BY clause is necessary

✡ All non-aggregate column names written in select must be written in GROUP BY.

✡ But some non-aggregate Attributes can be put in GROUP BY HAVING, that are not in the SELECT.

DEPT	No-of-Employes
ICP	2
ETA	3

*3 GROUP BY Multiple Columns

DEPT	MANAGER	No. of Employes
ICP	NULL	1
ETA	NULL	1
ETA	2	2

*4. There can be Multiple Aggregate function fields present in the select clause,

Correctly put all non-aggregates also in Group By

Q: Min and Max Salary of each dep.

> SELECT Dept, MIN(Salary), MAX(Salary)
FROM Employee GROUP BY Dept;

→ Last-third in functions..

* Aggregate function ~~with~~ GROUP BY nahi hi onde also nhi ho.

URBAN EDGE

* GROUP BY on NULLABLE Columns.

⇓

* Grouping will also take place on NULL values.

* NULL is also a group

Q: Count of Employees under each Manager

> SELECT Manager, COUNT(id) "No. of Emps"
FROM Employee
~~WHERE~~ GROUP BY Manager;

Manager	No. of EMPS
NULL	2
2	2
1	1

→ No. of Emps having no manager

* GROUP BY Try-Out

⇓

GROUP BY on MULTIPLE COLUMNS

Q: Display Dept, Manager and no. of Employees with the same designation working under each manager of each department

Each

Har department me Har Manager me under Har Designation me kitne Employee ha.

Mahadev Grouping.

> SELECT Dept, Manager, Designation ~~FROM~~, COUNT(Id)
FROM Employee
GROUP BY Dept, MANAGER, DESIGNATION;

★4. GROUP-BY – Quiz

URBAN EDGE

Q1. Errors

Jitne Select me ho utne saare GROUP BY me hone chahiye

⇩

Baat khatam

⇩

Nhi toh mismatch hoga Error ayega

Jitne SELECT me utne GROUP BY me ⇒ DISTINCT

toh toh Aggregate may be skipped

But in this question

```
> SELECT Make, Model
  FROM Computer
  GROUP BY Make;
```

↑ Model not here

Answer → expression not in aggregate or GROUP BY clause

Q2.

```
> SELECT Make
  FROM Employee
  GROUP BY Make;
```

Q3 =

Q4 =

Q5 ?? Finish theory.

* Exercise - GroupBy - locationwise Min Age
simple.

* Ex - GB - Recent PurchaseDate

↓ * Interesting [Tricky]

⇓

Recent Purchase Date → means → Largest / Maximum

⇓

Purchase Date

⇓

for a particular customer

> SELECT c.CustId, MAX (c.PurchaseDate)
FROM Invoice
GROUP BY c.CustId;

*3 Ex GB - WhiteProducts

⇓

Grouping on Multiple Columns.

↓ * to Filter White Products first → WHERE is used

*4. Ex GB - RecentPurchase ProductWise.

↓

group By both CustId and ProductId and the find the MAX (purchaseDate).

*5. Ex - GROUPBY - locationwise Min Age of Each Gender.

⇓

Arrange bhi karne ko bola.

6. HAVING

Need
Now, if we want another filtering on those GROUPS as well

↓

Use HAVING clause after GROUP BY.

Q: Find the departments in which total salary paid is greater than or equal to 90000.

```
> SELECT Dept, SUM(salary) FROM Employee
  GROUP BY Dept
  HAVING SUM(salary) >= 90000;
```

★ Difference b/w WHERE and HAVING.

WHERE:
To filter the result [based on Grouped Column]
- Use WHERE, instead of HAVING

Slow

```
> SELECT Dept, SUM(salary) FROM Employee GROUP BY Dept
  HAVING Dept <> 'SE';
```

```
> SELECT Dept, SUM(Salary) FROM Employee WHERE Dept<>'SE'
  GROUP BY Dept;
```

↓

Faster

HAVING:
★ To filter the results based on Aggregate Values of Grouped Columns

★ Aggregate functions / their values CANNOT be used in the WHERE clause.

INCORRECT QUERY:

```
>SELECT Dept, SUM(salary) From Employee
 Where SUM(salary) > 87000 Group By Dept
```

* Identify Mistakes

1) SELECT Manager, count(*) FROM Employee
GROUP BY Dept;

Manager rows will mismatch

Non-aggregate used with aggregate in SELECT must be used in the GROUP BY

2) Same

* Dos and Don't for GROUP BY - HAVING:

DOs

* GROUP BY → mandatory when a list of Non-aggregate and Aggregate in the SELECT statement.
* HAVING always after the GROUP by clause (written & executed)
* HAVING cannot come without GROUP BY.
* condition on Aggregate fun' always with HAVING Not with WHERE.

* * v. special must come

you can group some columns based on the Primary key.

DONT's

Imp

* GROUP BY should not have any Aggregate Columns.
  side na aggregate kita hai uske baare te Group By nhi X
* HAVING should not contain any Non-Aggregate columns that are not present in the GROUP BY clause
* Nested Aggregate Function Not allowed with the queries having GROUP BY

* Exe

— when N... Aggrega...

* Exercise - GB - Min Age Atleast 30

URBAN EDGE

> SELECT CustLocation "LOCATION", MIN(Age) "Min Age"
FROM ~~Employee~~ Customer
GROUP BY CustLocation
HAVING MIN(Age) >= 30;

* ALIAS cannot be used in Having clause for aggregate functions.

* Ex-GB- Loyal Customers

* Interesting

GROUP BY all 3 custId, VendorId, ProductId

COUNT (InvoiceId)

Kyuki us se Pata chalga actual count Kharidne ka

- When NO Aggregate function is there.

* Gr.

> Should not be used -> because this Leads to the Duplicate Values for those column values grouped.

Points

★ ALIAS cannot be used with HAVING clause.
given
for Aggregate function value

(Alias can be used in double quotes ~~if~~ was given)

URBAN EDGE

★ In normal ORDER BY you can Order By based on the column i.e not there in the SELECT clause.

But

★ Here while using it with GROUP BY you cannot use any other column to order that is not there in the SELECT clause.

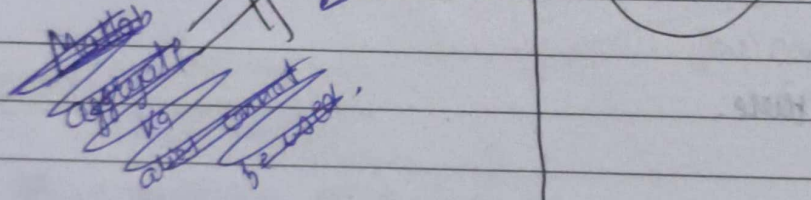
★ ALIAS names of columns can be used in ORDER BY.

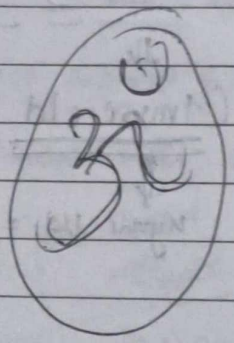
Imp

★★ Order by me Alias ✓

★ GROUP BY me ✓

★ HAVING ✗ kaise nahi





used in all cases then with order by you can use it in any case without " ".

↑ But ↓

★ Agar " " me sensitive use kardiya toh ORDER_BY me safe bhi ✓
& " use karo

* GROUP BY Column errors:

Errors

1) All the non-Aggregate Columns written in SELECT clause must be given in the GROUP BY clause;

ex1) > SELECT Dept, COUNT(id) AS "No. " FROM Employee GROUP BY DESIGNATION;

error: expression not in aggregate or GROUP BY columns.

ex2) SELECT Dept, Ename, COUNT(id) AS "No-of-Emp" FROM Employee GROUP BY Dept; ✗

error: expression not in aggregate or GROUP BY Colums.

2) WHERE clause cannot be used on Aggregate fields at all.

HAVING should be used after GROUP BY to apply conditions on aggregate fields.

#3 ORDER BY Along with GROUP BY

* Both Aggregate and Non-Aggregate Columns can be used to order but they must be in SELECT clause if GROUP BY is used.

✗ SELECT Dept, SUM(Salary) AS "Total_Salary" FROM Employee GROUP BY DEPT ORDER BY Designation.

↓ Correct

ORDER BY Dept; ✓

> SELECT DEPT, COUNT(*) AS "No-of-Employees" FROM Employee GROUP BY Dept ORDER BY No-of-Employees;

Double quotes used then → Case Sensitive
otherwise Not.

★ Quiz- GROUP BY

URBAN EDGE

Q1. Diff. b/w WHERE and ~~GROUP BY~~ HAVING when used with GROUP BY

Quiz- Group-By → Qus 4 | Doubt

Qus Where is used before Grouping and Having is used after grouping.

Q2. HAVING → Filter groups after grouping has been done.

Q3. ★ Q. Find only those model categories for which more than one computer exists.

Sol:

```
SELECT Model, COUNT(*), FROM computer
    GROUP BY Model
```

Q4. SELECT Model, COUNT(*)

Where vaali koi kahani nahi hai dhyaan naal such

If te doojo WHERE ch Aggregate use hi nhi kar skte.

Q5. If the dhyaan nahi WHERE lgaye.

→ Pehle hi WHERE price > 55000 krke vo vaale le liye then GROUP BY

mastery

Joins :

* Kabhi join karo vo cross-join karke hi result data aayega. Matlab, matchy attribute (key) dono column mein (based on one or more cols)

Introduction to Joins aayega

* SET OPERATIONS COMBINE DATA VERTICALLY but

Need :

* BUT HERE WE NEED HORIZONTAL MAPPINGS OF DATA FROM BOTH TABLES.

CompId	Model
1001	Vostro
1002	Precision

ID	EName	CompId
1	James Potter	1001
2	Ethan	Null

* Using set operations, we can find the CompIds allocated to employees.

* Comparable ya Same Columns ko hi UNION/INTERSECTION/Ex lete ha. VERTICAL

Intersection can be used

CompId
1001

⟸ CompIds allocated to Employees.

* But what if we need all the details of the Computers allocated to Employee.

↓ expected output

Id	Ename	CompId	Model
1	James	1001	Vostro

* Now, this data is scattered b/w / across tables how to combine them ??

⇓

[JOINS]

⇓

To MAP Data Horizontally b/w tables

" JOIN is AN SQL OPERATION, through which WE CAN COMBINE DATA FROM MULTIPLE TABLES BASED ON COMMON COLUMNS "

URBAN EDGE

=> Joins one of types → -> CROSS JOIN
-> INNER JOIN
-> SELF JOIN
-> OUTER JOIN (left, Right, Full)

1. # Fundamental Join → CROSS JOIN
↓ ↓
CARTESIAN PRODUCT

• Each row of first table combined with each row of second

> m rows in table A => CROSS JOIN
> n rows in table B ↓
Always produce

m x n rows

* RARELY USED as it produce a lot of meaningless data.
but base for other joins

Employee (Alias E)

ID	ENAME	DEPT	Compld
1	James	ICP	1001
2	Ethan	ETA	NULL
3	Emily	ETD	NULL

Computer (Alias C)

Compld	Make	Model	Year
1001	DELL	Vostro	2019
1002	DELL	[illegible]	[illegible]

rows = 3 * 2 = 6
columns = (8) (4+4)

> SELECT ID, EName, E.Compld As ECID, C.Compld As CCid, Model from Employee E
CROSS JOIN
Computer C;

Dhyaan Rakhna just

A×B / B×A

INNER JOIN

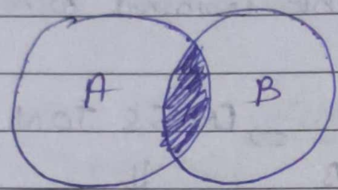
*2 INNER JOIN : (DETAILS Dono Taraf se Lene ke liye g common things among tables)

⇓

Most Frequently used JOIN.

" It matches the records from both the tables based upon the join condition and returns only the matching rows.

* (Every join starts from the CROSS JOIN and then the rows that do not meet condition are dropped)



ON <Condition>

7 SELECT ID, ENAME, E.COMPID, ECId, C.CompId CCId, Model FROM
EMPLOYEE Y INNER JOIN COMPUTER C
ON E.COMPId = C.COMPId

* Tarika

→ Ek row

match condition
each every row in Second
where exact matched take those rows.

* Try-out- Inner Join * V. V. Imp *

Not ~~true~~
0

URBAN EDGE

* NULL = NULL → UNKNOWN
  ↓
  ∴ NULL matching does not happen
  and rows with equivalent NULLs will not be included in the result

NULL values do not match!!

COMMON ERRORS (INNER JOIN)

1) Using comma separated table name with ON clause
   ↓
   Correct: SELECT ID, ENAME, E.compId FROM EMPLOYEE E INNER JOIN Computer C ON

* We can't use the Join condition in WHERE clause for INNER JOIN.
  ~~ERROR~~ [* but can do with CROSS JOIN]

★ INNER JOIN with Condition

To filter the combined rows on some condition

URBAN EDGE

Q) Fetch all employees from ETA who are assigned with a computer ??

Option 1:

```
SELECT Id, Fname, E.compId
AS E_compId, C.compId AS C_compId
Model
FROM Employee E INNER JOIN Computer C
ON E.compId = C.compId
WHERE EDept = 'ETA';
```

Fetches same as this join but we need computer details of these employees as well -> that's why join.

Two steps:

1) Join (to get all the rows and columns)
2) Filter is used on all rows of the result set

Option 2:

Combine Condition to Filter in ON with AND operator

-> Here the query is executed in single step as the query is executed condition to filter is applied right at the time of Join

```
Employee E. InnerJoin Computer C ON E.compId = C.compId
    AND Dept = 'ETA';
```

Order of Query Execution

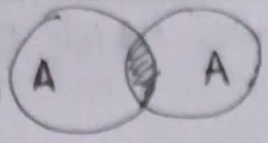
Identical results.

F W GB HI S D O L

From Join Where Group By HAVING Select Distinct Order

★3 SELF JOIN

(INNER JOIN only)



URBAN EDGE

→ "We combine a Table with itself based on Condition and Fetch Data"

→ Special Case of Inner Join where table is joined with itself.

Typical Employee - Manager (Self-Referencing Key)

EmpId ← ManagerId

CourseId ← Pre-requisite

Example : Employee

Id	Ename	- - - -	Manger

★ Manager colum is a SELF-REFERENCING FOREIGN key here. ⇓

It refers Id of the same table.

★ If we want EmpId, EmpName and employees' Mayer Id

↳ simple Query

→ Select EmpId, EmpName, ManagerId from Employee

BUT

If we want manger name also

⇓

SELF JOIN is required.

★ OUTER JOINS and its Types

URBAN EDGE

Outer Join :

↓

To fetch Matching Records of both the tables along with Non-Matching records from left, right or both sides.

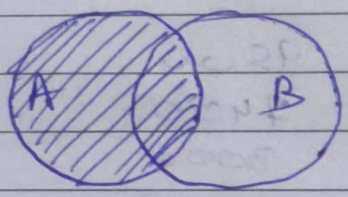
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN

1. LEFT OUTER JOIN :

⇓

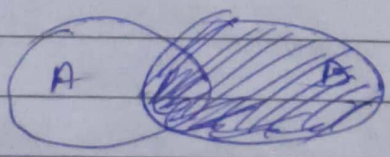
Fetches Rows that have matching values on the common attribute along with all records that are left in the LEFT TABLE

- If a row in the LEFT Table has no matching record in the Right Table → its corresponding columns in the result will contain the NULL values.



2. RIGHT OUTER JOIN :

- Matching + Right vaale un-matching also
- corresponding left values will be NULL



★3 FULL OUTER JOIN :

"Fetches records that have matching values on the common attribute along with all the records in Right and Left table.

Go see Animations if any Doubt

★1 LEFT OUTER JOIN :

Employee

ID	ENAME	DEPT	COMPID
1	James	ICP	1001
2	Ethan	ETA	NULL
3	Emily	ETA	NULL

Computer.

Compid	Make	Model	MYear
1001	Dell	Vostro	2013
1002	Dell	Precision	2014

"LEFT OUTER JOIN

Id	EName	Ecompid	CCompid	Model
1	James	1001	1001	Vostro
2	Ethan	NULL	NULL	NULL
3	Emily	NULL	NULL	NULL

> SELECT Id, Ename, E.Compid, C.Compid, C.Model
FROM Employee E LEFT OUTER JOIN Computer C
ON E.Compid = C.Compid;

★2. RIGHT OUTER JOIN :

> SELECT Id, Ename, E.Compid AS ECID, C.compid CCID, Model
FROM Employee E RIGHT OUTER JOIN Computer C
ON E.compid = C.Compid ;

Id	EName	ECompid	CCompid	Model
1	James	1001	1001	Vostro
NULL	NULL	NULL	1002	Precision

Mantra
Normal Left wale ko CROSS JOIN ki tarah hi match karte jao ek ek karke jaha-jaha match ho gaya left ka right me likho, Nhi to end me remaining Right ka add karo with corresponding NULLs on left.

* Left Outer Join me, naal hi oh jo vi se naal match nahi hoya last te ohnu ta ose vale paa to with NULL values on right.
* Right ch pehle left → right normal match karaya

*3. Jehre hoye match oh paate

Te last ch remaining Right valle paa te with NULLS on left.

*3. Full Outer JOIN:

(Algo.) Common Tarika use karo
↓
Right valle un-matched last ch check honge ten last ch hi aange.

ID	ENAME	DEPT	COMPID
1	James	ICP	1001
2	Ethan	ETA	NULL
3	Emily	ETA	NULL

CompId	Make	Model	MYear
1001	DELL	Vostro	2013
1002	DELL	Pres	2014

Id	Ename	ECompId	CompId	Model
1	James	1001	1001	Vostro
2	Ethan	NULL	NULL	NULL
3	Emily	NULL	NULL	NULL
Null	Null	Null	1002	Precision

(A + B) − (Matches)

TRYOUT.

JOIN

ON Must hai and

URBAN EDGE

LEFT OUTER JOIN WITH CONDITION :

* In INNER JOIN filters can be placed at two places:

-> 1) Using AND operator in ON clause along with joining criteria.

-> 2) In WHERE clause after Join operation is completed.

* Filtering Predicate with Outer Joins is not that simple

should be put carefully

1) - - Filter with outer JOIN using NULL conditions.

2) - - - Filtering in WHERE Clause V/s FILTERING IN ON clause.

3) - - - Filter Condition on main table attribute V/s Filter condition on Lookup table attribute.

V

Taking example of LEFT OUTER JOIN ONLY

★ LEFT OUTER JOIN with IS NULL / IS NOT NULL condition on Matching attribute

URBAN EDGE

In the Expression:

A LEFT OUTER JOIN B

A → Main table

B → LOOK UP TABLE

Computer

Compld	Make	Model	Mfgyear
1001	Dell	Vostro	2013
1002	Dell	Precision	2014
1003	Lenovo	Edge	2013
1004	Lenovo	Thinkpad	2014
1005	HP	Arizona	2015

Employee

Id	EName	Dept	compId
1	James	ICP	1001
2	Ethan	ETA	NULL
3	Emily	ETA	1002
4	Jack	ETA	NULL
5	Aya3	ICP	1003

Ccompla	Make	E.Id	E.CompId
1001	Dell	1	1001
1002	Dell	3	1002
1003	Lenovo	5	1003
1004	Lenovo	NULL	NULL
1005	HP	NULL	NULL

We know LEFT OUTER JOIN

a. IS NULL check on Matching attribute of the look up table.

```
SELECT C.Compld, C.Make, E.Id, E.Compld
FROM Computer C LEFT OUTER JOIN Employee E ON
C.Compld = E.Compld
WHERE E.Compla IS NULL
```

EXCEPT

⇓

Given Non-matching CompId records

b) IS NULL

⇓

N

OR

c) IS NOT

e) IS NO

b) IS NULL Check on matching attribute main table.

↓

No result

as Compld in Main Table is PK ∴ Not NULL.

c) ISO NOT NULL check on matching attribute of look up table

↓

results in Matched rows. ≡ (INNER JOIN)

d) IS NOT NULL check on matching attribute of main table.

↓

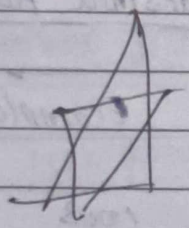
Saara is AA jayega

↓

Kyuki main table ki jo Voh PK hai na.

≡ LEFT OUTER JOIN without any condition.

E.Compld
1001
1002
1003
NULL
NULL



Saara Game EK ki PK or dusre ki FK se Ho rahi hai.

LEFT OUTER JOIN with condition on Main/ Look Up table

★ WHERE and ON clause

a) Filtering Predicate on Main Table in WHERE clause

```
SELECT C.CompId, C.Make, E.Id, E.compId
FROM Computer C LEFT OUTER JOIN Employee
ON C.CompId = E.comId
WHERE C.Make = 'Lenovo';
```

1) • First LEFT OUTER JOIN takes place

C → Last Page

2) Then WHERE Clause filters the desired rows.

CompId	Make	Eid	Ecompid
1003	Lenovo	5	1003
1005	Lenovo	NULL	NULL

b) Filtering Predicate on main table in ON clause.

1) It will match both Id and Make

1003	Lenovo	5	1003

CompId	Make	Eid	Ecompid
1001	Dell	NULL	NULL
1002	Dell	NULL	NULL
1003	Lenovo	5	1003
1004	Lenovo	NULL	NULL
1005	HP	NULL	NULL

Not some

Go according to upto Table

Check Two conditions Here

INNER JOIN AB (PK) CD (FK)

ON A = C AND B = D

URBAN EDGE

c) Filtering predicate on lookup table in WHERE Clause.

```
SELECT C.CompId, C.Make, E.Id, EmployeeId, E.CompId
FROM Computers C LEFT OUTER JOIN Employee
E ON C.CompId = E.CompId.
WHERE E.Dept = 'ETA';
```

1) First Left outer Join

CompId	Make	EId	ECompId	Dept
1001	Dell	1	1001	ICP
1002	Dell	3	1002	ETA
1003	Lenovo	5	1003	ICP
1004	Lenovo	NULL	NULL	NULL
1005	HP	NULL	NULL	NULL

2) then WHERE filter is performed

1002	Dell	3	1002	ETA

* Note! : Filtering Predicate on Lookup Table. (For Left Outer Join)

Using WHERE is of No USE as it behaves as INNER JOIN

d) Filtering Predicate on Lookup table in ON clause.

- Two conditions to be checked while doing LEFT OUTER JOIN
  - ★ if a Match found then add it
  - ★ otherwise just put from left and correspondingly NULLs on right.

Compid	Make	Eid	Ecompid	Dept
1001	Dell	NULL	NULL	NULL
1002	Dell	3	1002	A
1003	Lenovo	NULL	NULL	NULL
1004	Lenovo	NULL	NULL	NULL
1005	HP	NULL	NULL	NULL.

★ ON and WHERE ka same Result nhi aata.

* filter on main table should be written in where clause

URBAN EDGE

→ Main Table Attribute Condition → WHERE
as it filters the results after the join has been performed.

→ filter condition in Lookup table → AND clause with join condition in ON clause.

Why Lookup condition in ON as it takes LEFT outer join after applying the condition.

Imp!

→ No IF →

To filter (except for NULL check) using attribut from Lookup table will result in WHERE clause in wrong output

as all NULL Rows will get filtered and the purpose of using OUTER JOIN will get defeated.

If We want to conditionally fetch values from the Lookup table
then
additional criteria must be combined with join condition using AND operator.

* filter on main table should be written in where clause

URBAN EDGE

→ Main Table Attribute Condition → WHERE
as it filters the results after the join has been performed.

→ Filter condition in Lookup table → AND clause with join condition in ON clause.
Lookup condition

Why Lookup in ON
as it takes LEFT outer join after applying the condition.

Imp

NOTE →

To filter (except for NULL check) using attribut from Lookup table will result in WHERE clause in wrong output

as all NULL Rows will get filtered and the purpose of using OUTER JOIN will get defeated.

If We want to conditionally fetch values from the Lookup table
then
additional criteria must be combined with join condition using AND operator.

Ex: To ...
for only ...
manufact...

> SELEC...
AS ...
FROM ...
ON ...

#

EMP

Why 1) Main condition in WHERE

★ filter on main table should be written in where clause

URBAN EDGE

→ Main Table Attribute Condition → WHERE
as it filters the results after the join has been performed.

→ filter condition in Lookup table → AND clause with join condition in ON clause.

Why Lookup condition in ON as it takes LEFT outer join after applying the condition.

Imp

→ No IF →

To filter (except for NULL check) using attribut from Lookup table will result in WHERE clause in wrong output

as all NULL Rows will get filtered and the purpose of using OUTER JOIN will get defeated.

If We want to conditionally fetch values from the Lookup table then additional criteria must be combined with join condition using AND operator.

SET OPERATIONS: (Vertical mapping only - based on one / more cols)

URBAN EDGE

★ COMBINING DATA:

★ To fetch details from same or different tables.

→ Find the Computers that have been allocated to Employees.
→ Find ——— not ———

To get the details from multiple tables or from the same table in a single output

SET OPERATIONS

★ SET OPERATIONS CAN BE USED TO COMBINE OR COMPARE THE RESULTS OF TWO SELECT STATEMENTS

★ Performed on Two Tables having Same Degree (i.e same no. of Columns)

Set operations in MSQL:
→ UNION
→ UNION ALL
→ INTERSECT
→ EXCEPT

1. UNION / UNION ALL :

↓

- To combine the Results of 2/more SELECT Statements.
- SELECT may be on same or different tables.

★ V.V. Imp

- They must have same no. of columns
- And the data types of the columns at same position in the queries must be Compatible / or same colums (either same colums / same datatype / auto convertible)

★ • UNION removes all duplicates from the result.

Two Records are considered duplicates if values at correspondig positions of all their columns match

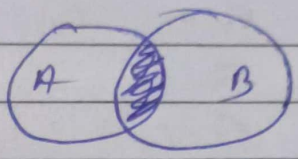
#2. INTERSECT
Fetches the Common (DISTINCT) elements returned by both the participating queries.

(Boolean conditions come
↳ same degree
↳ same type of corresponding columns.

* To find the computer Ids that have been allocated to employees.

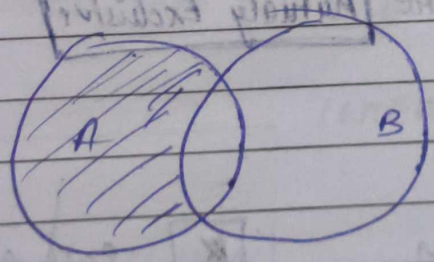
> SELECT Compld from Computer C
INTERSECT
SELECT Compld from Employee E

Compld
1001
1002
1003



#3. EXCEPT (A-B)
↳ finds those Elements returned by First Query that do not exist in the elements returned by Second query.

" to find the Computer Ids that have NOT been allocated"



$A - B = (A \cup B) - B$

Compld
1004
1005