SELECT - list function

FROM - indicate the tables or views

WHERE - Conditions of what rows included

GROUP BY - indicate categorization of results

HAVING - indicate conditions under which a category will be included.

ORDER BY- sorts the result

Processing order: E.G - **SELECT** c1, c2 **FROM** t1 **WHERE** [condition on rows] **GROUP BY** c1, c2 **HAVING** [conditions on groups] **ORDER BY** c1;

**DISTINCT** - eliminate duplicate rows

E.G - **SELECT DISTINCT**(c1) **FROM** t1;

**AS** - alternative column header name

E.G - **SELECT** X **AS** Y,

**FROM** t1

**WHERE**[condition]

X - original name, Y - alternative name

Operator for WHERE:

**BETWEEN** - **WHERE** X **BETWEEN** n **AND** m

AND - T if all conditions T

OR - T if either conditions T

ANY - T if any of subquery values meet the condition

EXISTS - T if the subquery returns one or more records

NOT - displays a record if the conditions not T

IN - T if the operand is equal to one of a list of expressions

LIKE - compare strings

E.G - **SELECT** c1 **FROM** t1 **WHERE** c1 **LIKE** ‘%abc’;

List items that end with abc

‘abc%’ start with abc

‘a%c’ start with a, end with c

‘\_a%’ contain a on second place

‘a\_\_%’ at least 3 digits

**IS NOT NULL**

E.G - **SELECT** \* **FROM** t1 **WHERE** c1 **IS NOT NULL**

**IN** - to indicate certain value

E.G - **SELECT** c1, c2, c3 **FROM** t1 **WHERE** c1 **IN** (‘x’, ‘y’,’z’);

(from table 1, select c1, c2, c3 when c1 is x, y or z)

< > not equal to

GROUP BY:

**AVG** -

**SUM** -

**MIN** -

**MAX** -

**COUNT** -

E.G - **SELECT** c1, **COUNT**(c1) **FROM** t1 **GROUP BY** c1;

E.G - **SELECT** c1, **COUNT**(c1) **FROM** t1 GROUP **BY** c1 **HAVING COUNT**(c1) > 1;

(having after group by work on group condition)

INNER JOIN- match primary and foreign key

E.G - **SELECT** \*

**FROM** t1

**INNER JOIN** t2 **ON** t1.c1 = t2.c2

**INNER JOIN** t3 **ON** t2.c2 = t3.c3

**INNER JOIN** t4 **ON** t3.c3 = t4.c4

**WHERE** [condition]

(c1= pk in t1, c2= pk in t2)

NATURAL JOIN- duplicate columns is eliminated in the result table

E.G - **SELECT** c1,c2 **FROM** t1 **NATURAL JOIN** t2

CROSS JOIN - cross product

LEFT OUTER JOIN - produces **a complete set of records from table A**

E.G - **SELECT** \* **FROM** t1 **LEFT OUTER JOIN** t2 **ON** t1.c1 = t2.c2

A set of records **only in table A** but not in table B

E.G - **SELECT** \* **FROM** t1 **LEFT OUTER JOIN** t2 **ON** t1.c1 = t2.c2 **WHERE** t2.pk **IS NULL**

FULL OUTER JOIN - includes all columns from each table

E.G - **SELECT** \* **FROM** t1 **FULL OUTER JOIN** t2 **ON** t1.c1 = t2.c2

SELF JOIN - require aliases

E.G - **SELECT** E.EmployeeID, E.EmployeeName, M.EmployeeName **AS** Manager

**FROM** Employee E, Employee M

**WHERE** E.EmployeeSupervisor=M.EmployeeID

length() - length of characters

limit - limit the display output

E.G - **SELECT** city, **length**(city) from station

**ORDER BY** **length**(city),city asc

**limit** 1;

**SELECT** city, **length**(city) from station

**ORDER BY** **length**(city) desc

**limit** 1

**MYSQL REGEXP Example**

E.G - **SELECT DISTINCT** city **FROM** station **WHERE** city **REGEXP** "^[aeiou].\*"

1. Match beginning of string(^): Gives all the names **starting** with ‘sa’.

E.G - sam,samarth.

**SELECT** name **FROM** student\_tbl **WHERE** name **REGEXP** '^sa';

1. Match the end of a string($): Gives all the names **ending** with ‘on’.

E.G - norton,merton.

**SELECT** name **FROM** student\_tbl **WHERE** name **REGEXP** 'on$';

1. Match **zero or one** instance of the strings preceding it(?): Gives all the titles **containing** ‘com’.

E.G - comedy , romantic comedy.

**SELECT** title **FROM** movies\_tbl **WHERE** title **REGEXP** 'com?';

1. Matches **any** of the patterns p1, p2, or p3(p1|p2|p3): Gives all the names **containing** ‘be’ or ‘ae’.

E.G - Abel, Baer.

**SELECT** name **FROM** student\_tbl **WHERE** name **REGEXP** 'be|ae' ;

1. Matches **any** character listed between the square brackets([abc]): Gives all the names **containing** ‘j’ or ‘z’.

E.G - Lorentz, Rajs.

**SELECT** name **FROM** student\_tbl **WHERE** name **REGEXP** '[jz]' ;

1. Matches any lower case letter between ‘a’ to ‘z’- ([a-z]) ([a-z] and (.)): Retrieve all names that contain a letter in the range of ‘b’ and ‘g’, followed by any character, followed by the letter ‘a’.

E.G - Tobias, sewall.

Matches any single character(.)

**SELECT** name **FROM** student\_tbl **WHERE** name **REGEXP** '[b-g].[a]' ;

1. Matches any character **not** listed between the square brackets.([^abc]): Gives all the names not containing ‘j’ or ‘z’.

E.G - nerton, sewall.

**SELECT** name **FROM** student\_tbl **WHERE** name **REGEXP** '[^jz]' ;

1. Matches the end of words[[:>:]]: Gives all the titles ending with character “ack”.

E.G - Black.

**SELECT** title **FROM** movies\_tbl **WHERE REGEXP** 'ack[[:>:]]';

1. Matches the **beginning of words**[[:<:]]: Gives all the titles starting with character “for”.

E.G - Forgetting Sarah Marshal.

**SELECT** title **FROM** movies\_tbl **WHERE** title **REGEXP** '[[:<:]]for';

1. Matches a character class[:class:]: i.e [:lower:]- lowercase character ,[:digit:] – digit characters etc. Gives all the titles containing alphabetic character only.

E.G - stranger things, Avengers.

**SELECT** title **FROM** movies\_tbl **WHERE REGEXP** '[:alpha:]' ;

**MYSQL ORDER Example**

ASC - acending

DESC - decending

LEFT - extract char from left

RIGHT - extract char from right

E.G - **SELECT** name **FROM** students **WHERE** marks > 75 **ORDER BY** **RIGHT**(name,3), id **ASC**

**Example:**

1. List pizzas with the substring 'i' anywhere within the pizza name.

**SELECT** pizza **FROM** menu **WHERE** pizza **LIKE** '%i%'

1. List all pizzas, giving pizza name, price and country of origin where the country of origin has NOT been recorded (i.e. is missing).

**SELECT** pizza, price, country **FROM** menu **WHERE** country **IS NULL**

1. Give the most expensive pizzas from each country of origin.

**SELECT** country, **MAX**(price) **FROM** menu **WHERE** country **IS NOT NULL** **GROUP BY** country

1. Give the average price of pizzas from each country of origin, do not list countries with only one pizza.

**SELECT** country, **AVG**(price) **AS** avg **FROM** menu **WHERE** country **IS NOT NULL** **GROUP BY** country **HAVING COUNT**(\*) > 1

1. List all 'fish' ingredients used in pizzas, also list the pizza names. Do not use a subquery.

**SELECT** items.ingredient, recipe.pizza **FROM** items **NATURAL JOIN** recipe **WHERE** items.type='fish'

1. List pizzas with at least one 'meat' ingredient.You must use a subquery.

**SELECT** pizza **FROM** recipe **WHERE** recipe.ingredient **IN** (**SELECT** items.ingredient **FROM** items **WHERE** type='meat') **GROUP BY** pizza **ORDER BY** pizza

1. List all ingredients for the Italian pizza (i.e. country = 'italy'). You must use a subquery.

**SELECT** ingredient **FROM** recipe, menu **WHERE** country **IN** ('italy') **AND** recipe.pizza=menu.pizza **GROUP BY** ingredient **ORDER BY** ingredient

1. Give pizzas and prices for pizzas that are more expensive than all Italian pizzas. You must use a subquery.

**SELECT** pizza, price **FROM** menu **WHERE** price>(select max(price) **FROM** menu **WHERE** country='italy')

1. List all pizzas that cost less than 'siciliano' pizza, also give their prices.

**SELECT** pizza, price **FROM** menu **WHERE** price<(**SELECT**  price **FROM** menu **WHERE** pizza='siciliano')

1. List each ingredient and the pizza that contains the largest amount of this ingredient.

**SELECT** recipe.ingredient, recipe.pizza, recipe.amount

**FROM** recipe

**INNER JOIN**

(**SELECT** **MAX**(amount) **AS** amt, ingredient **FROM** recipe

**GROUP BY** ingredient) **AS** rcp

**ON** recipe.ingredient = rcp.ingredient

**AND** recipe.amount = rcp.amt

| 1. List ingredients used in more than one pizza.   **SELECT** ingredient **FROM** recipe **GROUP BY** ingredient **HAVING COUNT**(\*)>1 **ORDER BY** ingredient |
| --- |

1. List ingredients that are not used in any pizza.

**SELECT** ingredient **FROM** items **WHERE** ingredient **NOT IN** (**SELECT** ingredient **FROM** recipe)

1. Give all pizzas that originate from the same country as the 'siciliano' pizza.

**SELECT** pizza **FROM** menu **WHERE** country=(**SELECT** country **FROM** menu **WHERE** pizza like '%siciliano%') and pizza **< >** 'siciliano'

1. Find the pizza which uses the largest number of ingredients.

**SELECT** pizza **FROM** recipe **GROUP BY** pizza **HAVING COUNT**(pizza)>=**ALL** (**SELECT** **COUNT**(pizza) **FROM** recipe **GROUP BY** pizza)