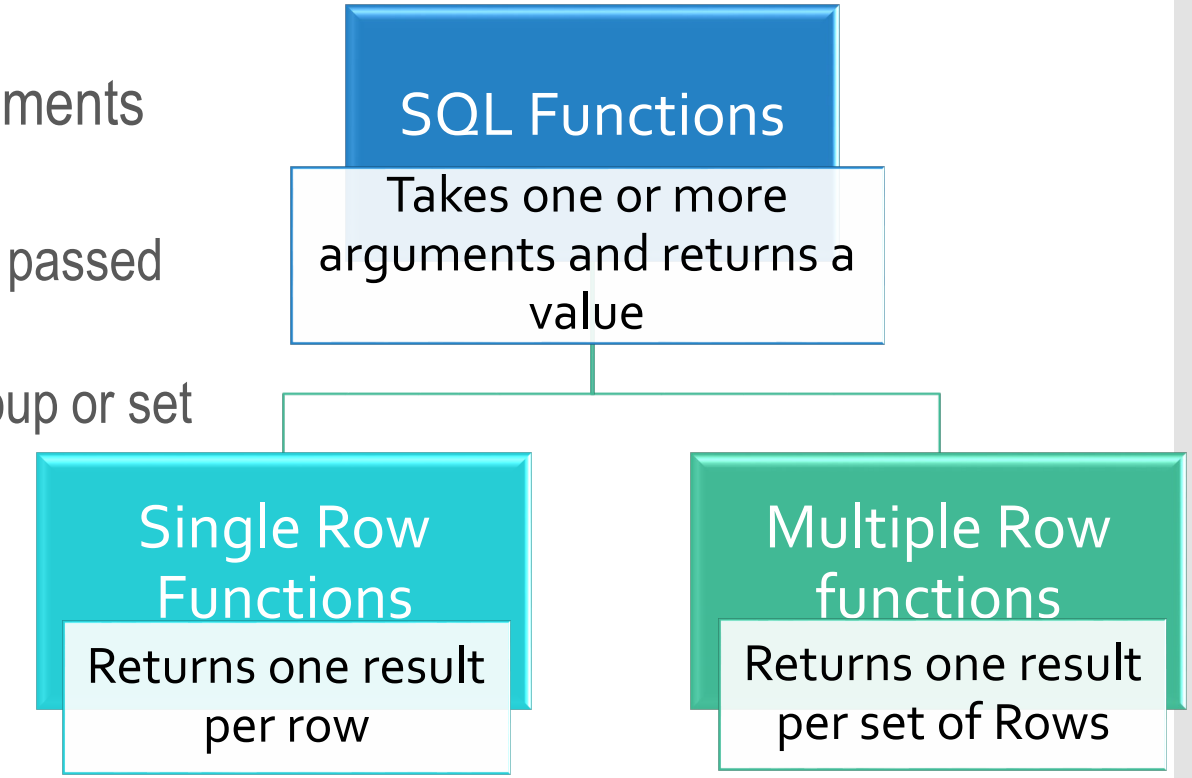
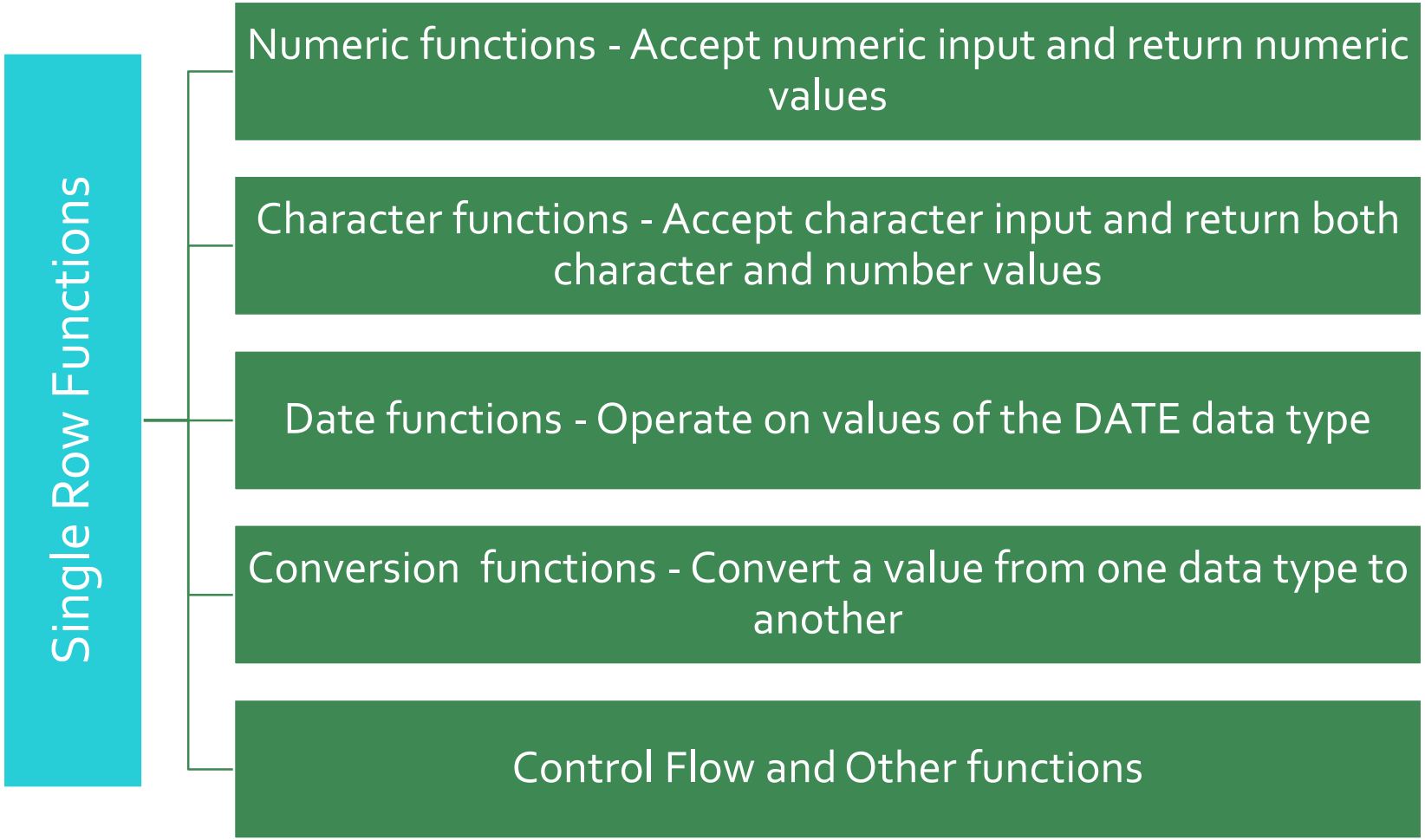
SQL Functions

FUNCTIONS IN SQL

- Required to transform the retrieved column values
- For e.g. rounding off numbers, upper / lower case, date formatting, etc.
- Used in expressions, calculations in SQL statements
- Two categories:
  - Single-row function: Applied on columns that are passed as parameters to it
  - Multi-row or Aggregate function: Applied on a group or set of rows



TYPES OF SINGLE ROW FUNCTIONS



greatlearning
Learning for Life

NUMERIC FUNCTIONS

- ROUND: Rounds a numeric value to a specified decimal
- TRUNC: Truncates a numeric value to a specified decimal
- FLOOR: Nearest integer less than the given number
- CEIL: Nearest integer more than the given number

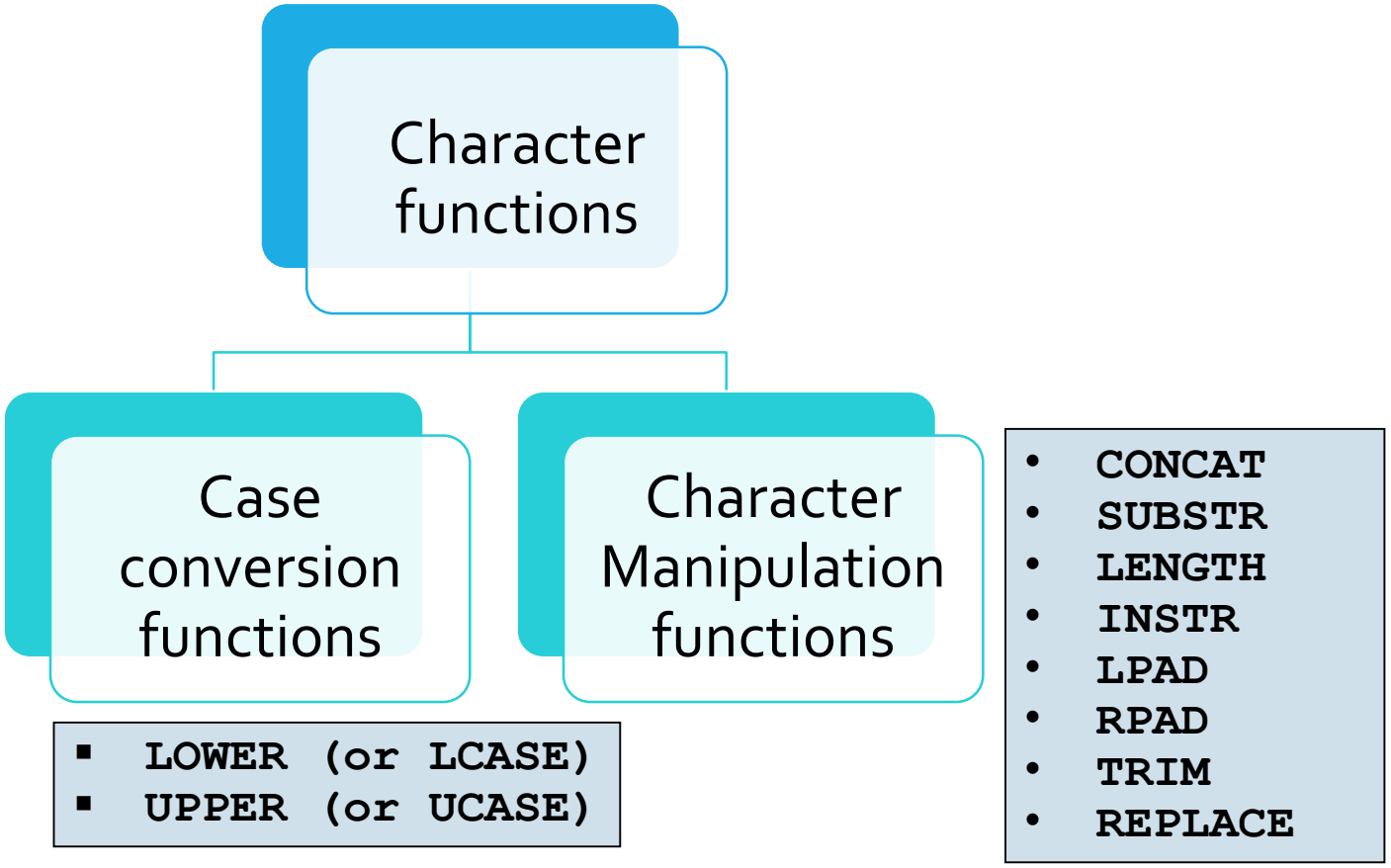
```
SELECT ROUND(45.923,2), ROUND(45.923,0),
ROUND(45.923,-1);

SELECT TRUNC(45.923,2), TRUNC(45.923),
TRUNC(45.923,-1);

SELECT last_name, salary, MOD(salary, 5000)
FROM   Employees
WHERE  job_id = 'SA_REP';
```

Refer to https://dev.mysql.com/doc/refman/8.0/en/functions.html for a complete list of MySQL single-row functions

TYPES OF CHARACTER FUNCTIONS



CASE CONVERSION FUNCTIONS

- Though MySQL string search is case insensitive ('abc' = 'Abc' = 'ABC'), it is advised to keep the case of strings same in both sides while comparing
- To make it case insensitive search:

```
SELECT employee_id, last_name, department_id
FROM   Employees
WHERE  LOWER(last_name) = 'higgins';
```

- Above condition can also be written as:

```
WHERE UPPER(last_name) = 'HIGGINS';
```

greatlearning
Learning for Life

CHARACTER MANIPULATION FUNCTIONS

Function	Result	Explanation
`CONCAT('Hello', 'World')`	`HelloWorld`	Joins values together (only use two parameters with CONCAT.)
`SUBSTR('HelloWorld',1,5)`	`Hello`	Extracts a string of determined length
`LENGTH('HelloWorld')`	`10`	Shows the length of a string as a numeric value
`INSTR('HelloWorld', 'W')`	`6`	Finds the numeric position of a named character
`LPAD(salary,10,'*')`	`*****24000`	Returns an expression left-padded to the length of *n* characters with a character expression
`RPAD(salary, 10, '*')`	`24000*****`	Returns an expression right-padded to the length of *n* characters with a character expression
`REPLACE ('JACK and JUE','J','BL')`	`BLACK and BLUE`	Returns an expression right-padded to the length of *n* characters with a character expression
`TRIM('H' FROM 'HelloWorld')`	`elloWorld`	Trims leading or trailing characters (or both) from a character string

CHARACTER MANIPULATION FUNCTIONS

- One query can contain multiple single-row functions in it:

```
SELECT employee_id,
CONCAT(first_name, ' ', last_name) NAME, job_id,
LENGTH (last_name),
INSTR(last_name, 'a') "Contains 'a'?"
FROM   Employees
WHERE  SUBSTR(job_id, 4) = 'REP'
AND INSTR(last_name, 'a') > 0;
```

WORKING WITH DATES

- Date values should be enclosed in single-quotes while using them in a query:

```
SELECT last_name, hire_date
FROM   Employees
WHERE  hire_date < '1988-02-01';
```

- Database stores dates as numbers, it performs calculations using arithmetic operators such as addition and subtraction.
- Query to display the no. of weeks the employee has been employed for all employees in department 90

```
SELECT last_name, hire_date, DATEDIFF(CURDATE(),
hire_date)/7 AS Weeks
FROM   Employees
WHERE  department_id = 90;
```

DATE FUNCTIONS

Function	Result

`CURRENT_DATE()(or CURDATE())`
`CURRENT_TIME() (or CURTIME())`
`CURRENT_TIMESTAMP() (or NOW())`
	Return current date / time / timestamp

`PERIOD_DIFF(yymm1, yymm2)`	Finds the number of months between two periods.
`ADDDATE(date, n) or DATE_ADD` `SUBDATE(date, n) or DATE_SUB`	Adds / subtracts +/-*n* number of days or interval specified to/from *date*.
`LAST_DAY(date)`	Finds the date of the last day of the month that contains *date*

Refer to https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html for complete list of date & time functions

DATE FUNCTIONS

Examples:

```
SELECT CURRENT_DATE(), CURRENT_TIME();
```

```
SELECT PERIOD_DIFF(201810, 201801);
```

```
SELECT employee_id, first_name, hire_date,
PERIOD_DIFF(DATE_FORMAT(curdate(), '%Y%m'),
DATE_FORMAT(hire_date, '%Y%m'))/12 as yrs_of_service
FROM Employees WHERE department_id = 90;
```

```
SELECT hire_date, ADDDATE(hire_date, 100) AS
After100days, SUBDATE(hire_date, INTERVAL 1 MONTH) AS
Before1Month FROM Employees WHERE department_id = 90;
```

```
SELECT LAST_DAY(CURDATE());
```

CONVERSION FUNCTIONS

- Data in one form may have to converted to another form before comparing them
- For example, a number '1000' in character format will have to converted before it can be compared with another number
- Some of them are implicitly converted and some of them have to explicitly converted
- MySQL automatically or implicitly converts below data types provided they have valid data
- For e.g. a character 'A' cannot be converted to equivalent number, nor a character-based date, '99-99-9999' can be converted to equivalent date

From	To
VARCHAR or CHAR	NUMBER
VARCHAR or CHAR	DATE
NUMBER	VARCHAR or CHAR
DATE	VARCHAR or CHAR

DATE_FORMAT: DATE TO CHAR

- DATE_FORMAT converts a datetime data type to a value of VARCHAR data type in the format specified by the format_specifier
- A format specifier is a character literal that describes the format of datetime stored in a character string

Specifier	Description
%a	Abbreviated weekday name (Sun..Sat)
%b	Abbreviated month name (Jan..Dec)
%c	Month, numeric (0..12)
%D	Day of the month with English suffix (0th, 1st, 2nd, 3rd, ...)
%d	Day of the month, numeric (00..31)
%H or %h	Hour (00..23) or (01 .. 12)
%i	Minutes, numeric (00..59)
%j	Day of year (001..366)
%k or %l	Hour (0..23) or (1..12)

Specifier	Description
%M	Month name (January..December)
%m	Month, numeric (00..12)
%p	AM or PM
%r	Time, 12-hour (hh:mm:ss followed by AM or PM)
%S or %s	Seconds (00..59)
%T	Time, 24-hour (hh:mm:ss)
%U or %u	Week (00..53), where Sunday is the first day of the week; WEEK() mode 0
%V or %v	Week (01..53), where Sunday is the first day of the week; WEEK() mode 2; used with %X
%W	Weekday name (Sunday..Saturday)
%Y or %y	Year numeric four or two digits

DATE_FORMAT: DATE TO CHAR

- Examples:
  - `SELECT DATE_FORMAT(curdate(),'%d-%m-%Y');`
  - `SELECT CONCAT(first_name, ' joined on ', DATE_FORMAT(hire_date, '%W, %M %D, %Y')) AS emp_desc`
  - `FROM Employees WHERE employee_id = 123;`
  - `SELECT CONCAT('Today is ', DATE_FORMAT(curdate(), '%W, %D of %M, %Y')) AS Today;`
  - `SELECT CONCAT('Now it''s ', DATE_FORMAT(now(), '%h:%i:%s %p'));`

OTHER DATA TYPE CONVERSION METHODS

- Numbers are implicitly converted to CHAR in MySQL.
- `E.g. SELECT CONCAT('Olympics-', 2019);`
- Any of the following functions can be used for explicit conversions:
  - `CAST()`
  - `CONVERT()`
- Examples:
  - `SELECT CAST('2018-10-31' AS DATE);`
  - `SELECT CONVERT('2018-10-31', DATE);`
  - `SELECT CAST(150 AS CHAR);`
  - `SELECT CONVERT(150, CHAR);`
  - `SELECT CAST('15:06:10' AS TIME);`
  - `SELECT CONVERT('15:06:10', TIME);`

CONTROL FLOW FUNCTIONS

- Functions that control the flow of the query based on data value:
  - IF(expr1, expr2, expr3) - If expr1 is True, return expr2, otherwise expr3
  - IFNULL(expr1, expr2) – If expr1 is not NULL, return expr1, otherwise expr2
  - NULLIF(expr1, expr2) – If expr1 = expr2, return NULL, otherwise expr1
  - CASE value WHEN condition THEN result WHEN condition THEN result … ELSE result END - To simulate IF-ELSE ladder in SQL

CONTROL FLOW FUNCTIONS

Examples:

```
SELECT IF(1=2, 'Y', 'N');  -- Returns 'N'
SELECT salary, IF(salary > 10000, 'High', 'Low') AS sal_grade
FROM Employees WHERE employee_id = 123;

SELECT IFNULL(NULL, 100);  -- Returns 100
SELECT employee_id, first_name, department_id, salary,
IFNULL(commission_pct, 0)
FROM Employees WHERE department_id IN (80, 90) AND salary >
10000;

SELECT NULLIF(10, 20);     -- Returns 10
SELECT first_name, last_name FROM Employees WHERE
NULLIF(LEFT(first_name, 1), LEFT(last_name, 1)) IS NULL;
```

greatlearning
Learning for Life

Analytic Functions

ANALYTIC FUNCTIONS

- Analytic functions calculate an aggregate value based on a group of rows
- However unlike aggregate functions, analytic functions return multiple rows for each group.
- Can be used to compute moving averages, running totals, percentages or top-N results within a group
- Syntax:

```
Select …from table analytic_function([ arguments ])
OVER (( [PARTITION BY <...>] [ORDER BY <....>]
[<window_clause>] )
```

- Some examples: CORR, CUME_DIST, LAG, LEAD, LISTAGG, NTH_VALUE, RANK, DENSE_RANK, etc.
- Aggregate functions like AVG, SUM, etc. also can be used as analytic functions

ANALYTIC FUNCTIONS: BASIC AGGREGATE FUNCTIONS

- COUNT

```
SELECT last_name, department_id,
COUNT(*) OVER (PARTITION BY department_id) dept_cnt
FROM Employees
WHERE department_id IN (30, 40);
```

- SUM

```
SELECT employee_id, last_name, SUM(salary)
OVER (PARTITION BY department_id) dept_total, department_id
FROM Employees
WHERE department_id IN (30, 40);
```

- AVG

```
SELECT employee_id, last_name, AVG(salary)
OVER (PARTITION BY department_id) dept_Avg,department_id
FROM Employees
WHERE department_id IN (30, 40);
```

greatlearning
Learning for Life

ANALYTIC FUNCTIONS

- ROW_NUMBER():- Gives a running serial number to a partition of records
- RANK():- Calculates the rank of a value in a group of values. The return type is NUMBER.
- DENSE_RANK():- Acts like the RANK function except that it assigns consecutive ranks
- LEAD computes an expression based on the next rows
- i.e. rows coming after the current row and return value to current row
- LEAD (expr, offset, default)
  - expr = expression to compute from leading row
  - offset = index of the leading row relative to the current row
  - default = value to return if the <offset> points to a row beyond partition range
- FIRST_VALUE returns the first result from an ordered set.
- Refer to analytic_func.sql

ANALYTIC FUNCTIONS

Other analytic functions:

- CUME_DIST - display the cumulative distribution, or the relative position in the set, of each of the employees, as well as all the original data.
- NTILE - Breaks a result set into a specified number of approximately equal groups, or buckets, rows permitting.
  - If the no. of rows in the set is smaller than the number of buckets specified, the number of buckets will be reduced so there is one row per bucket.
- PERCENT_RANK - Assigns value between 0-1 which represents the position of the current row relative to the set as a percentage
- Refer to analytic_func.sql

SUMMARY

- Single row functions accept one or more arguments and return one result per row.
- Conversion functions are used to convert one data type to another with a specified format.
- Group functions accept one or more arguments and return one result for multiple rows
- Different types of joins are used for retrieving data from multiple tables
- Set operations on tables are UNION and INTERSECT
- Analytic functions or windowing functions work similar to group functions
- Analytic functions do not reduce the no. of rows in the result unlike group functions

greatlearning
Learning for Life

Dimension Functions

greatlearning
Learning for Life

DIMENSION FUNCTIONS

- In data warehouse, a dimension is a collection of reference information about a measurable event
- The events are referred as facts and are stored in fact tables
- For example, dimension tables store data like Customer details, Product details, Sales information, etc.
- Fact tables contain the foreign key to each of these dimension tables with the corresponding measure say, total sales, average sales, spend, etc.
- Functions like rollup, cube, etc. are required in the data warehouse context to aggregate the data in the fact table levels

greatlearning
Learning for Life

ROLLUP

- A simple extension to the GROUP BY clause.
- Enables a SELECT statement to calculate multiple levels of subtotals across a specified group of dimensions.
- Also calculates a grand total.
- Highly efficient, adding minimal overhead to a query, easy to use.

```
SELECT department_id, SUM(salary)
FROM Employees
WHERE department_id IS NOT NULL
GROUP BY department_id WITH ROLLUP;

SELECT department_id,job_id, sum(salary)
FROM Employees
WHERE department_id IS NOT NULL
GROUP BY department_id, job_id WITH ROLLUP;
```

GROUPING SETS

- GROUPING SETS are a further extension of the GROUP BY clause .
- It lets you specify multiple groupings of data
- The output of ROLLUP includes the rows produced by the regular GROUP BY operation along with the summary rows
- GROUPING SETS used to generate summary information at the level you choose without including all the rows produced by the regular GROUP BY operation.

```
SELECT department_id,job_id, SUM(salary),
GROUPING(department_id), GROUPING(job_id)
FROM Employees
GROUP BY department_id, job_id WITH ROLLUP;
```

greatlearning
Learning for Life

THANK YOU