Lecture 3 Notes

SQL Queries

Discussion Points:

- SQL: SELECT Statement
- SQL: Restricting Data
- SQL: Sorting Data
- SQL: Single-Row Functions

❖ SQL: SELECT Statement

- *Writing SQL Statements*
  - SQL statements are not case sensitive.
  - SQL statements can be on one or more lines.
  - Keywords cannot be abbreviated or split across lines.
  - Clauses are usually placed on separate lines.
  - Indents are used to enhance readability.

- *Basic Select Statement*

  Example:
  ```
  SELECT *|{[DISTINCT] column|expression [alias],...}
  FROM table;
  ```

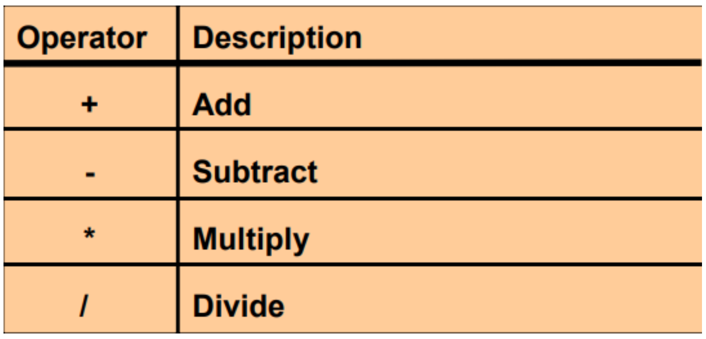
- *Selecting All Columns*

  Example:
  ```
  SELECT *
  FROM departments;
  ```

- *Selecting Specific Columns*

  Example:
  ```
  SELECT department_id, location_id
  FROM departments;
  ```

- *Arithmetic Expressions*



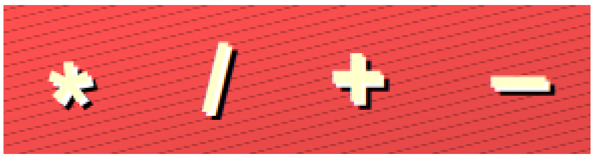
Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide

- *Using Arithmetic Expressions*

  Example:
  ```
  SELECT last_name, salary, salary + 300
  FROM employees;
  ```

- *Operator Precedence*
  - Multiplication and division take priority over addition and subtraction.
  - Operators of the same priority are evaluated from left to right.

  - Parentheses are used to force prioritized evaluation and to clarify statements.

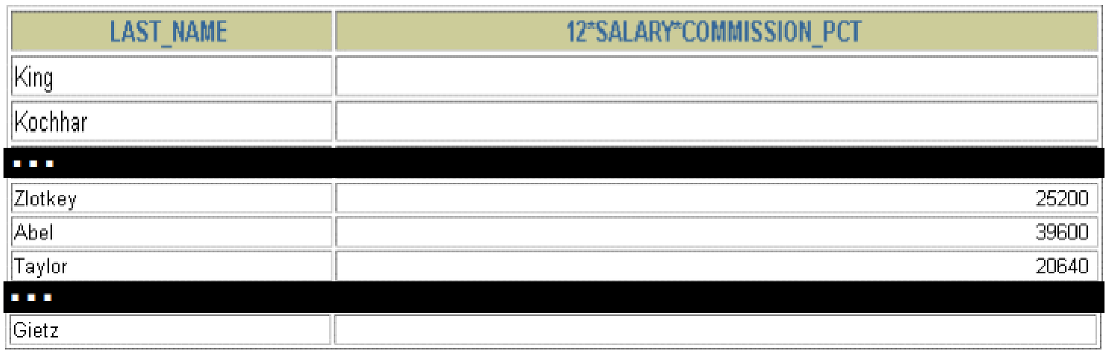


Example:
SELECT last_name, salary, 12*salary+100
FROM employees;

Example:
SELECT last_name, salary, 12*(salary+100)
FROM employees;

- *NULL Values*
  - A null is a value that is unavailable, unassigned, unknown, or inapplicable.
  - A null is not the same as zero or a blank space.
  - Arithmetic expressions containing a null value evaluate to null.

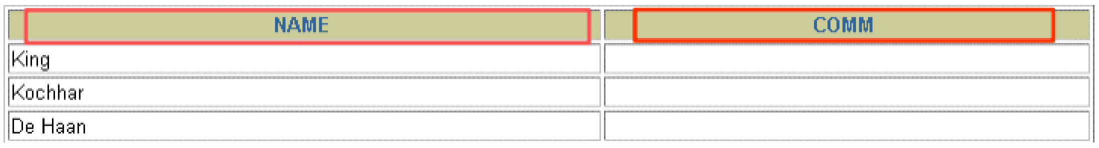
Example:
SELECT last_name, 12*salary*commission_pct
FROM employees;



LAST_NAME	12*SALARY*COMMISSION_PCT
King	
Kochhar	
...	
Zlotkey	25200
Abel	39600
Taylor	20640
...	
Gietz	

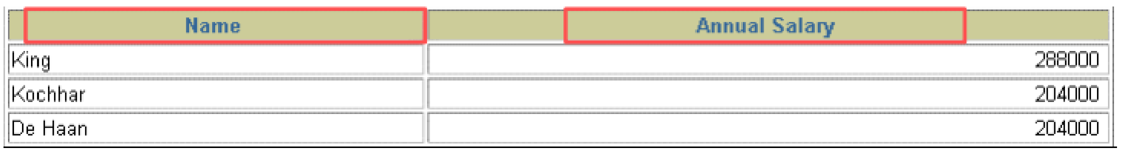
- *A column alias:*
  - Renames a column heading.
  - Is useful with calculations.
  - Immediately follows the column name - there can also be the optional AS keyword between the column name and alias.
  - Requires double quotation marks if it contains spaces or special characters or is case sensitive.

Example:
SELECT last_name AS name, commission_pct comm FROM employees;



NAME	COMM
King	
Kochhar	
De Haan	

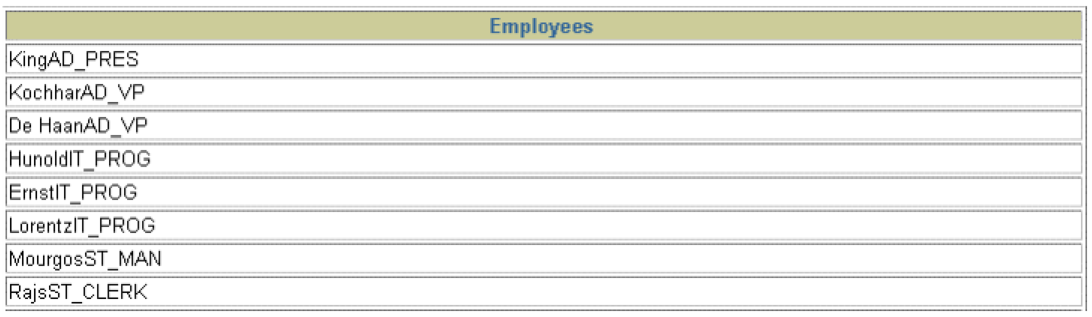
Example:
SELECT last_name "Name", salary*12 "Annual Salary"
FROM employees;



Name	Annual Salary
King	288000
Kochhar	204000
De Haan	204000

- *A concatenation operator:*
    - Concatenates columns or character strings to other columns.
    - Is represented by two vertical bars (||).
    - Creates a resultant column that is a character expression.

Example:
SELECT last_name||job_id AS "Employees"
FROM employees;



Employees
KingAD_PRES
KochharAD_VP
De HaanAD_VP
HunoldIT_PROG
ErnstIT_PROG
LorentzIT_PROG
MourgosST_MAN
RajsST_CLERK

- *Literal Character Strings*
    - A literal is a character, a number, or a date included in the SELECT list.
    - Date and character literal values must be enclosed within single quotation marks.
    - Each character string is output once for each row returned.

Example:
SELECT last_name ||' is a '||job_id
AS "Employee Details"
FROM employees;

- *Duplicate Rows:*
    - The default display of queries is all rows, including duplicate rows.
    - Eliminating Duplicate Rows

  Example:
  SELECT DISTINCT department_id
  FROM employees;

❖ SQL: Restricting Data

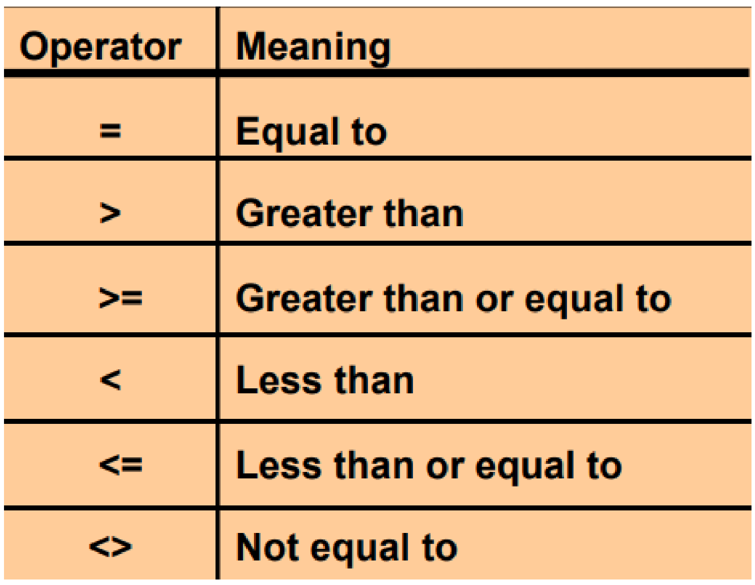
- *Restrict the rows returned by using the WHERE clause.*

  *SELECT *|{[DISTINCT] column|expression [alias],...}*
  FROM table
  [WHERE condition(s)];

  Example:
  SELECT employee_id, last_name, job_id, department_id
  FROM employees
  WHERE department_id = 90 ;

- *Character / Date Values*
    - Character strings and date values are enclosed in single quotation marks.
    - Character values are case sensitive, and date values are format sensitive.
    - The default date format is DD-MON-RR.

  Example:
  SELECT last_name, job_id, department_id
  FROM employees
  WHERE last_name = 'Whalen';



Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

- *Using Comparison Condtitions*
  Example:
  ```
  SELECT last_name, salary
  FROM employees
  WHERE salary <= 3000;
  ```

- *Other Comparison Conditions*

Operator	Meaning
BETWEEN ...AND...	Between two values (inclusive),
IN(set)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

- *Use the BETWEEN condition to display rows based on a range of values.*

  Example:
  ```
  SELECT last_name, salary
  FROM employees
  WHERE salary BETWEEN 2500 AND 3500;
  ```

- *Use the IN membership condition to test for values in a list.*

  Example:
  ```
  SELECT employee_id, last_name, salary, manager_id
  FROM employees
  WHERE manager_id IN (100, 101, 201);
  ```

- *Use the LIKE condition to perform wildcard searches of valid search string values.*
  - Search conditions can contain either literal characters or numbers:
    - % denotes zero or many characters.
    - _ denotes one character.

  Example:
  ```
  SELECT first_name
  FROM employees
  WHERE first_name LIKE 'S%';
  ```

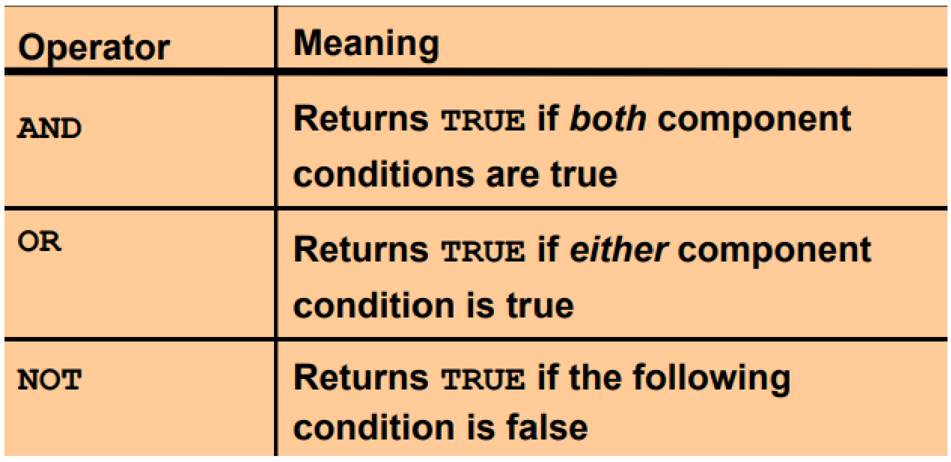
- *Pattern Matching*

  Example:
  ```
  SELECT last_name
  FROM employees
  WHERE last_name LIKE '_o%';
  ```

- *Using NULL Conditions*

  Example:
  ```
  SELECT last_name, manager_id
  FROM employees
  WHERE manager_id IS NULL;
  ```

- *Logical Conditions*



Operator	Meaning
AND	Returns TRUE if *both* component conditions are true
OR	Returns TRUE if *either* component condition is true
NOT	Returns TRUE if the following condition is false

- *AND Example*

  Example:
  ```
  SELECT employee_id, last_name, job_id, salary
  FROM employees
  WHERE salary >=10000
  AND job_id LIKE '%MAN%';
  ```

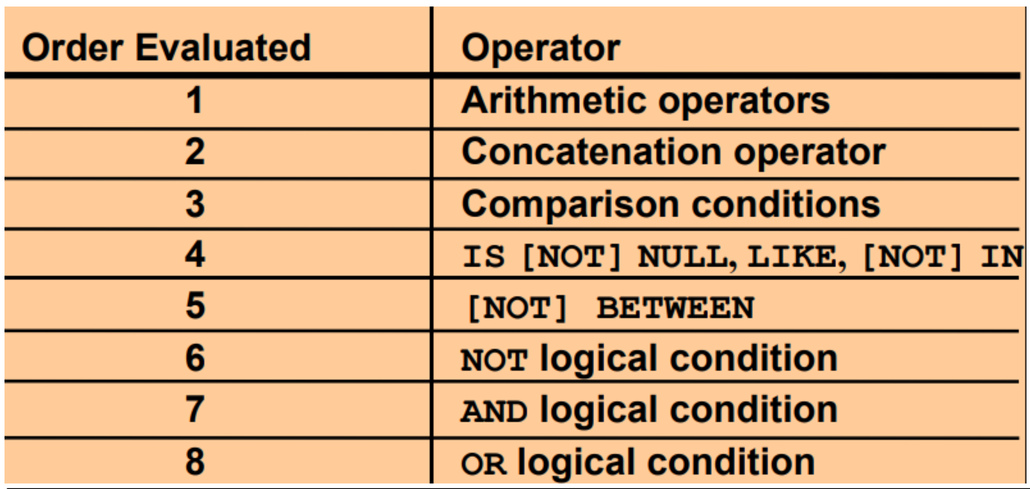
- *OR Example*

  Example:
  ```
  SELECT employee_id, last_name, job_id, salary
  FROM employees
  WHERE salary >= 10000
  OR job_id LIKE '%MAN%';
  ```

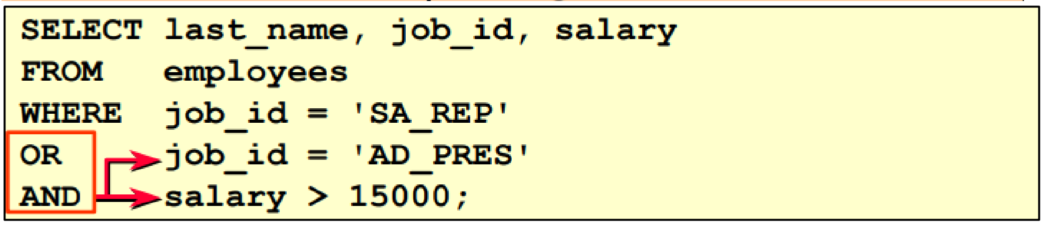
- *NOT Example*

  Example:
  SELECT last_name, job_id
  FROM employees
  WHERE job_id
  NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP');

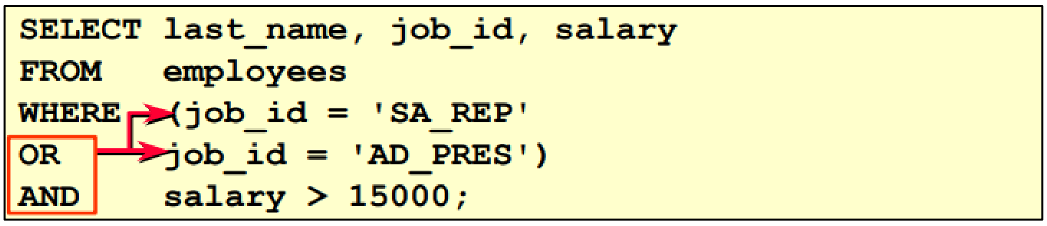
- *Rules of Precedence*



Order Evaluated	Operator
1	Arithmetic operators
2	Concatenation operator
3	Comparison conditions
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	NOT logical condition
7	AND logical condition
8	OR logical condition



```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id = 'SA_REP'
OR     job_id = 'AD_PRES'
AND    salary > 15000;
```



```
SELECT last_name, job_id, salary
FROM   employees
WHERE  (job_id = 'SA_REP'
OR     job_id = 'AD_PRES')
AND    salary > 15000;
```

❖ SQL: Sorting Data

- *ORDER BY Clause*
  - Sort rows with the ORDER BY clause
    - ASC: ascending order, default
    - DESC: descending order
  - The ORDER BY clause comes last in the SELECT statement.

*SELECT *|{[DISTINCT] column|expression [alias],...}*
FROM table
[WHERE condition(s)]
[ORDER BY {column, expr, alias} [ASC|DESC]];

- *Sorting by Ascending Order*

  Example:
  SELECT last_name, job_id, department_id, hire_date
  FROM employees
  ORDER BY hire_date ;

- *Sorting by Descending Order*

  Example:
  SELECT last_name, job_id, department_id, hire_date
  FROM employees
  ORDER BY hire_date DESC ;

- *Sorting by Column Alias*

  Example:
  SELECT employee_id, last_name, salary*12 annsal
  FROM employees
  ORDER BY annsal;

- *Sorting by Multiple Columns*
  - You can sort by a column that is not in the SELECT list.

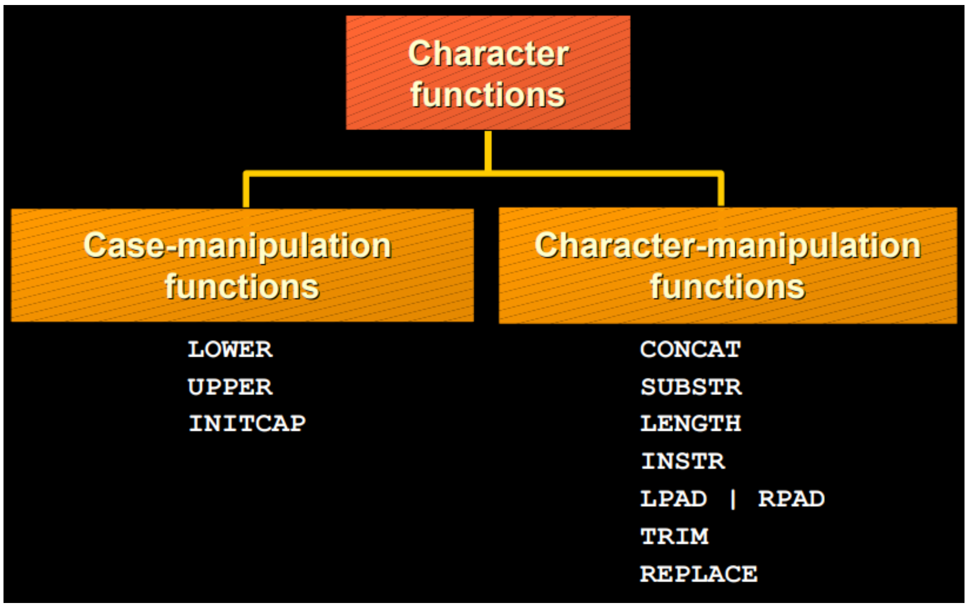
  Example:
  SELECT last_name, department_id, salary
  FROM employees
  ORDER BY department_id, salary DESC;

❖ SQL: Single-Row Functions

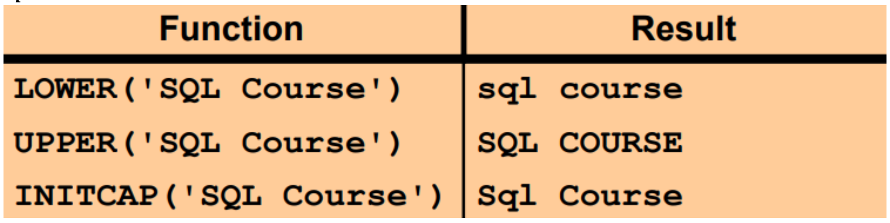
- *Single Row Functions*
  - Manipulate data items
  - Accept arguments and return one value
  - Act on each row returned
  - Return one result per row
  - May modify the data type
  - Can be nested
  - Accept arguments which can be a column or an expression

- *Single Row Function Types:*
  - General
  - Character
  - Number
  - Date
  - Conversion

- *Character Functions*



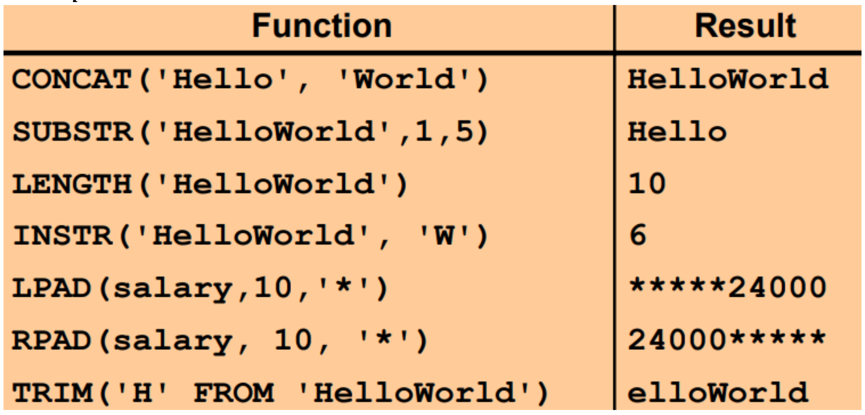
- *Case Manipulation Functions*



Function	Result
`LOWER('SQL Course')`	`sql course`
`UPPER('SQL Course')`	`SQL COURSE`
`INITCAP('SQL Course')`	`Sql Course`

Example:
SELECT employee_id, last_name, department_id
FROM employees
WHERE LOWER(last_name) = 'higgins';

- *Character-Manipulation Functions*

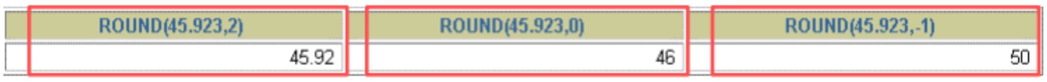


Function	Result
`CONCAT('Hello', 'World')`	`HelloWorld`
`SUBSTR('HelloWorld',1,5)`	`Hello`
`LENGTH('HelloWorld')`	`10`
`INSTR('HelloWorld', 'W')`	`6`
`LPAD(salary,10,'*')`	`*****24000`
`RPAD(salary, 10, '*')`	`24000*****`
`TRIM('H' FROM 'HelloWorld')`	`elloWorld`

Example:
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';

- *Number Functions*
  - *ROUND:* Rounds value to specified decimal
    ROUND(45.926, 2) -> 45.93

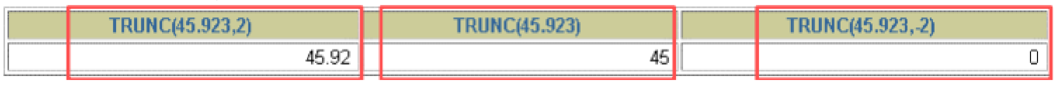
    Example:
    SELECT ROUND(45.923,2), ROUND(45.923,0),
    ROUND(45.923,-1)
    FROM DUAL;



ROUND(45.923,2)	ROUND(45.923,0)	ROUND(45.923,-1)
45.92	46	50

  - *TRUNC:* Truncates value to specified decimal
    TRUNC(45.926, 2) -> 45.92

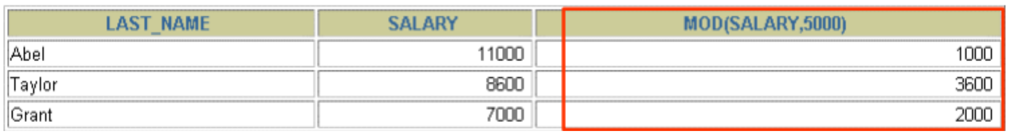
    Example:
    SELECT TRUNC(45.923,2), TRUNC(45.923),
    TRUNC(45.923,-2)
    FROM DUAL;



TRUNC(45.923,2)	TRUNC(45.923)	TRUNC(45.923,-2)
45.92	45	0

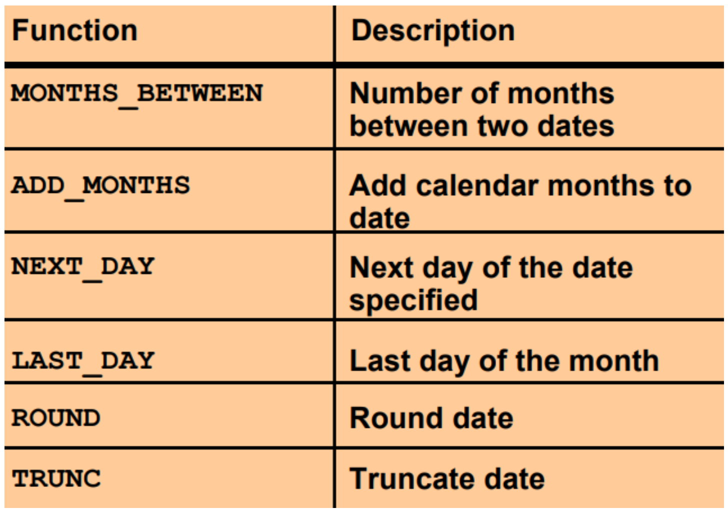
  - *MOD:* Returns remainder of division
    MOD(1600, 300) -> 100

    Example:
    SELECT last_name, salary, MOD(salary, 5000)
    FROM employees
    WHERE job_id = 'SA_REP';



LAST_NAME	SALARY	MOD(SALARY,5000)
Abel	11000	1000
Taylor	8600	3600
Grant	7000	2000

- *Date Functions*



Function	Description
MONTHS_BETWEEN	Number of months between two dates
ADD_MONTHS	Add calendar months to date
NEXT_DAY	Next day of the date specified
LAST_DAY	Last day of the month
ROUND	Round date
TRUNC	Truncate date

Examples:

- MONTHS_BETWEEN ('01-SEP-95','11-JAN-94') -> 19.6774194
- ADD_MONTHS ('11-JAN-94',6) -> '11-JUL-94'
- NEXT_DAY ('01-SEP-95','FRIDAY') -> '08-SEP-95'
- LAST_DAY('01-FEB-95') -> '28-FEB-95'

- Assume SYSDATE = '25-JUL-95':
- ROUND(SYSDATE,'MONTH') -> 01-AUG-95
- ROUND(SYSDATE ,'YEAR') -> 01-JAN-96
- TRUNC(SYSDATE ,'MONTH') -> 01-JUL-95
- TRUNC(SYSDATE ,'YEAR') -> 01-JAN-95

- Arithmetic with Date
  - Add or subtract a number to or from a date for a resultant date value.
  - Subtract two dates to find the number of days between those dates.
  - Add hours to a date by dividing the number of hours by 24.

  Example:
  SELECT last_name, (SYSDATE-hire_date)/7 AS WEEKS
  FROM employees
  WHERE department_id = 90;

❖ **References:**

- "Database System Concepts", Avi Silberschatz, Henry F. Korth, S. Sudarshan, McGraw-Hill.
- "Database Management Systems", Raghu Ramakrishnan, Johannes Gehrke, McGraw-Hill.
- "Fundamentals of Database Systems", R. Elmasri, S. B. Navathe, Pearson.
- Oracle SQL Resources
- Other Internet Sources