AGENDA

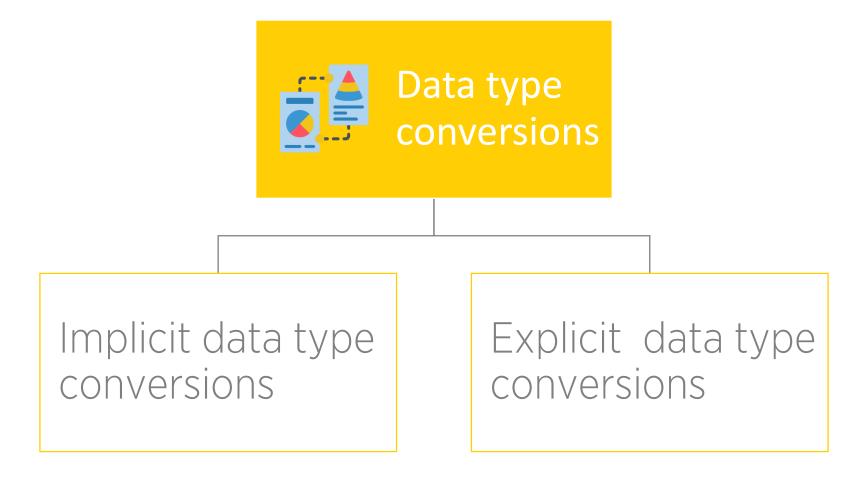
- 1. SQL Basic Functions
- 2. Aggregate Functions
- 3. Group by & Having Clause



SQL Basic Functions



Conversion Functions



Implicit Data Type Conversions

- o A VARCHAR2 or CHAR value can be implicitly converted to NUMBER or DATE type value by Oracle. Similarly, a NUMBER or DATA type value can be automatically converted to character data by Oracle server. Note that the implicit interconversion happens only when the character represents the valid number or date type value respectively.
- o For example, examine the below SELECT queries. Both the queries will give the same result because Oracle internally treats 15000 and '15000' as same.

```
Query-1
SELECT employee_id,first_name,salary
FROM employees
WHERE salary > 15000;

Query-2
SELECT employee_id,first_name,salary
FROM employees
WHERE salary > '15000';
```

Explicit Data Type Conversions

- SQL Conversion functions are single row functions which are capable of typecasting column value, literal or an expression. TO_CHAR, TO_NUMBER and TO_DATE are the three functions which perform cross modification of data types.
- TO_CHAR function is used to typecast a numeric or date input to character type with a format model (optional).

```
SELECT first_name, TO_CHAR (hire_date, 'MM, DD,
YYYY') HIRE_DATE, TO_CHAR (salary, '$99999.99')
Salary
FROM employees
WHERE rownum < 5;</pre>
```



| | ♦ FIRST_NAME | ♦ HIRE_DATE | SALARY |
|---|--------------|--------------|------------|
| 1 | Steven | 06, 17, 2003 | \$24000.00 |
| 2 | Neena | 09, 21, 2005 | \$17000.00 |
| 3 | Lex | 01, 13, 2001 | \$17000.00 |
| 4 | Alexander | 01, 03, 2006 | \$9000.00 |

Explicit Data Type Conversions

The TO NUMBER function converts a character value to a numeric datatype. If the string being converted contains nonnumeric characters, the function returns an error. The SELECT gueries below accept numbers as character inputs and prints them following the format specifier

```
SELECT TO NUMBER ('121.23', '9G999D99')
FROM DUAL;
TO NUMBER ('121.23', '9G999D99')
121.23
SELECT TO NUMBER ('1210.73', '9999.99')
FROM DUAL;
TO NUMBER ('1210.73', '9999.99')
1210.73
```

TO_DATE function

 The function takes character values as input and returns formatted date equivalent of the same.

NLS_DATE_LANGUAGE: is an expression that specifies the language for day and month names in the string.

```
SELECT
TO_DATE( '5 Jan 2017', 'DD MON YYYY')
FROM dual;
```

Functions - String Functions

| Function | Example | Result | Purpose |
|----------------|--|------------|---|
| CONCAT | CONCAT('A','BC') | 'ABC' | Concatenate two strings and return the combined string. |
| CONVERT | CONVERT('Ä Ê Í', 'US7ASCII', 'WE8ISO8859P1') | 'A E I' | Convert a character string from one character set to another. |
| <u>INITCAP</u> | INITCAP('hi there') | 'Hi There' | Converts the first character in each word in a specified string to uppercase and the rest to lowercase. |
| <u>INSTR</u> | INSTR('This is a playlist', 'is') | 3 | Search for a substring and return the location of the substring in a string |
| <u>LENGTH</u> | LENGTH('ABC') | 3 | Return the number of characters (or length) of a specified string |

String Functions

| Function | Example | Result | Purpose |
|----------------|---------------------------------------|-------------------|--|
| <u>LOWER</u> | LOWER('Abc') | 'abc' | Return a string with all characters converted to lowercase. |
| <u>LPAD</u> | LPAD('ABC',5,'*') | '**ABC' | Return a string that is left-padded with the specified characters to a certain length. |
| <u>LTRIM</u> | LTRIM(' ABC ') | 'ABC ' | Remove spaces or other specified characters in a set from the left end of a string. |
| <u>REPLACE</u> | REPLACE('JACK AND JOND','J','BL'); | 'BLACK AND BLOND' | Replace all occurrences of a substring by another substring in a string. |

String Functions

| Function | Example | Result | Purpose |
|---------------|--------------------------------------|----------|--|
| <u>RPAD</u> | RPAD('ABC',5,'*') | 'ABC**' | Return a string that is right-padded with the specified characters to a certain length. |
| RTRIM | RTRIM(' ABC ') | ' ABC' | Remove all spaces or specified character in a set from the right end of a string. |
| <u>SUBSTR</u> | SUBSTR('Oracle Substring', 1, 6) | 'Oracle' | Extract a substring from a string. |
| <u>TRIM</u> | TRIM(' ABC ') | 'ABC' | Remove the space character or other specified characters either from the start or end of a string. |
| <u>UPPER</u> | UPPER('Abc') | 'ABC' | Convert all characters in a specified string to uppercase. |

Functions - Comparison Functions

- COALESCE show you how to substitute null with a more meaningful alternative.
- DECODE learn how to add if-then-else logic to a SQL query.
- NVL return the first argument if it is not null, otherwise, returns the second argument.
- NVL2 show you how to substitute a null value with various options.
- NULLIF return a null if the first argument equals the second one, otherwise, returns the first argument.
- CASE The Oracle CASE statement has the functionality of an IF-THEN-ELSE statement. Starting in Oracle 9i, you can use the CASE statement within a SQL statement.

Oracle NVL(), NVL2() Function Examples

The following example returns N/A because the first argument is null:

```
1 SELECT
2 NVL(NULL, 'N/A')
3 FROM
4 dual;
```

The following statement returns two because the first argument is null.

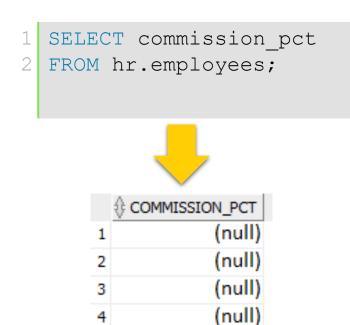
```
1 SELECT
2 NVL2(NULL, 1, 2) -- 2
3 FROM
4 dual;
```

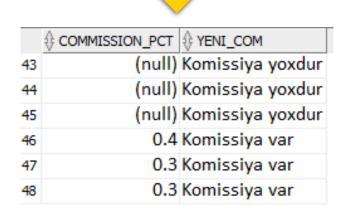
 The following example returns the second argument which is the ABC string because the first argument is not null.

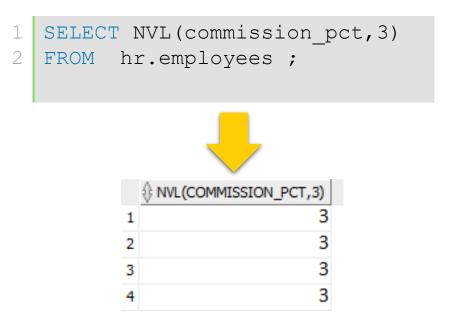
```
1 SELECT
2 NVL2(1, 'ABC', 'XYZ')
3 FROM
4 dual;
```

Oracle NVL(), NVL2() Function Examples

```
SELECT commission_pct, NVL2(commission_pct, 'Komissiya var', 'Komissiya yoxdur') AS yeni_com FROM hr.employees;
```







NVL2() Function Example

SELECT commission_pct,
salary, manager_id FROM
hr.employees;



| | | | MANAGER_ID |
|----|--------|-------|------------|
| 1 | (null) | 24000 | (null) |
| 2 | (null) | 17000 | 100 |
| 3 | (null) | 17000 | 100 |
| 45 | (null) | 2500 | 124 |
| 46 | 0.4 | 14000 | 100 |
| 47 | 0.3 | 13500 | 100 |
| 80 | 0.1 | 6200 | 149 |
| 81 | (null) | 3200 | 120 |
| 82 | (null) | 3100 | 120 |

SELECT NVL2 (commission_pct, salary, manager_id) as new_col FROM hr.employees;



| | NEW_COL |
|----|---------|
| 1 | (null |
| 2 | 100 |
| 3 | 100 |
| 45 | 12 |
| 46 | 1400 |
| 47 | 1350 |
| 80 | 620 |
| 81 | 12 |
| 82 | 12 |
| | |

Oracle COALESCE

 The Oracle COALESCE () function accepts a list of arguments and returns the first one that evaluates to a non-null value. The following example returns one because it is the first non-null argument:

```
1 SELECT
2 COALESCE(NULL,1) -- return 1
3 FROM
4 dual;
```

The following example returns null because all arguments are null:

```
SELECT
COALESCE(NULL, NULL) -- return NULL
FROM
dual;
```

Oracle COALESCE Function Example

```
1 SELECT phone_number, email, city, COALESCE(phone_number, email, city)
2 AS new_cl
FROM details;
```

| phone_number | email | city | new_cl |
|--------------|---------------|----------|---------------|
| 0551112233 | (null) | Baku | 0551112233 |
| (null) | ali@gmail.com | Goranboy | ali@gmail.com |
| (null) | (null) | Lankaran | Lankaran |

Oracle DECODE() Function

- The Oracle DECODE() function allows you to add the procedural if-then-else logic to the query.
- o In the following example, the Oracle DECODE() function compares the first argument (1) with the second argument (1). Because they are equal, the function returns the second argument which is the string One. The second example is slightly different from the one above. The query returns a null value because one does not equal two.

```
1 SELECT
2 DECODE(1, 1, 'One') -- return 'One'
3 FROM
4 dual;
```

```
1 SELECT
2 DECODE(1, 2, 'One') -- return Null
3 FROM
4 dual;
```

o If you want to specify a default value when the first argument is not equal to the second one, you append the default value to the argument list as shown below:

```
1 SELECT
2 DECODE(1, 2, 'One','Not one') -- return 'Not one'
3 FROM
4 dual;
```

Oracle DECODE() Function Example

```
SELECT first_name, DECODE (substr(first_name, 1, 1), 'A', 'Ilk herf
A-dir', substr(first_name, 1, 1)) AS new_1
FROM hr.employees;
```



| NEW_1 |
|----------------|
| Е |
| S |
| М |
| D |
| н |
| S |
| Ilk herf A-dır |
| |

Oracle CASE() Function

The Oracle CASE statement has the functionality of an IF-THEN-ELSE statement. Starting in Oracle 9i, you can use the CASE statement within a SQL statement.

```
CASE [ expression ]
WHEN condition 1 THEN result 1
WHEN condition 2 THEN result 2 ...
WHEN condition n THEN result n
ELSE result END
```

DATA SCIENCE ACADEMY

Oracle CASE() Function Example

```
SELECT first_name, salary, CASE
WHEN salary between 0 and 3000 THEN 'Kasibdir'
WHEN salary between 3001 and 8000 THEN 'Kasib deyil'
WHEN salary between 8001 and 10000 THEN 'Varlidir'
ELSE 'Chox yuksek gelirlidir'
END isci_yasayis
FROM hr.employees;
```



| FIRST_NAME | SALARY | ISCI_YASAYIS |
|------------|--------|------------------------|
| Steven | 24000 | Chox yuksek gelirlidir |
| Neena | 17000 | Chox yuksek gelirlidir |
| Lex | 17000 | Chox yuksek gelirlidir |
| Alexander | 9000 | Varlidir |
| Bruce | 6000 | Kasib deyil |
| David | 4800 | Kasib deyil |

Oracle NULLIF() Function

- The Oracle NULLIF() function accepts two arguments. It returns a null value if the two
 arguments are equal. In case the arguments are not equal, the NULLIF() function
 returns the first argument.
- o For example, the following statement returns a null value because the first argument equals the second one.

```
1 SELECT
2 NULLIF(100,100) -- null
3 FROM
4 dual;
```

 However, the following example returns the first value (100) because the two arguments are different:

```
1 SELECT
2 NULLIF(100,200) - 100
3 FROM
4 dual;
```

Functions - Date Functions

| Function | Example | Result | Purpose |
|-------------------|--|--|---|
| ADD_MONTHS | ADD_MONTHS(DATE '2016-02-29', 1) | 31-MAR-16 | Add a number of months (n) to a date and return the same day which is n of months away. |
| CURRENT_DATE | SELECT CURRENT_DATE FROM dual | 06-AUG-2017 19:43:44 | Return the current date and time in the session time zone |
| CURRENT_TIMESTAMP | SELECT CURRENT_TIMESTAMP FROM dual | 06-AUG-17 08.26.52.742000000 PM -07:00 | Return the current date and time with time zone in the session time zone |
| DBTIMEZONE | SELECT DBTIMEZONE FROM dual; | -07:00 | Get the current database time zone |
| EXTRACT | EXTRACT(YEAR FROM SYSDATE) | 2017 | Extract a value of a date time field e.g., YEAR, MONTH, DAY, from a date time value. |

Date Functions

| Function | Example | Result | Purpose |
|----------------|---|--|--|
| LAST_DAY | LAST_DAY(DATE '2016-02-01') | 2016-02-29 | LAST_DAY returns the date of the last day of the month that contains date. |
| LOCALTIMESTAMP | SELECT LOCALTIMESTAMP FROM dual | 06-AUG-17 08.26.52.742000 000 PM | Return a TIMESTAMP value that represents the current date and time in the session time zone. |
| MONTHS_BETWEEN | MONTHS_BETWEEN(DATE '2017-07-01', DATE '2017-01-01') | 6 | Return the number of months between two dates. |
| NEW_TIME | NEW_TIME(TO_DATE('08-07-2017 01:30:45', 'MM-DD-YYYY HH24:MI:SS'), 'AST', 'PST') | 06-AUG-2017 21:30:45 | Convert a date in one time zone to another |
| NEXT_DAY | NEXT_DAY(DATE '2000-01-01', 'SUNDAY') | 02-JAN-00 | Get the first weekday that is later than a specified date. |
| ROUND | ROUND(DATE '2017-07-16', 'MM') | 01-AUG-17 | Return a date rounded to a specific unit of measure. |

DATA SCIENCE ACADEMY

Date Functions

| Function | Example | Result | Purpose |
|--------------|--|---|--|
| SYSDATE | SYSDATE | 01-AUG-17 | Return the current system date and time of the operating system where the Oracle Database resides. |
| SYSTIMESTAMP | SELECT SYSTIMESTAMP FROM dual; | 01-AUG-17 01.33.57.9290000 00 PM -07:00 | Return the system date and time that includes fractional seconds and time zone. |
| TO_CHAR | TO_CHAR(DATE'2017- 01-01', 'DL') | Sunday, January 01, 2017 | Convert a <u>DATE</u> or an <u>INTERVAL</u> value to a character string in a specified format. |
| TO_DATE | TO_DATE('01 Jan 2017', 'DD MON YYYY') | 01-JAN-17 | Convert a date which is in the character string to a DATE value. |
| TRUNC | TRUNC(DATE '2017-07- 16', 'MM') | 01-JUL-17 | Return a date truncated to a specific unit of measure. |
| TZ_OFFSET | TZ_OFFSET('Europe/London') | +01:00 | Get time zone offset of a time zone name from UTC |

Aggregate functions



Aggregate Functions

- o Oracle functions calculate on a group of rows and return a single value for each group.
- We commonly use the aggregate functions together with the GROUP BY clause. The GROUP BY clause divides the rows into groups and an aggregate function calculates and returns a single result for each group.
- o If you use aggregate functions without a GROUP BY clause, then the aggregate functions apply to all rows of the queried tables or views.

AVERAGE()

 Oracle AVG() syntax The Oracle AVG() function accepts a list of values and returns the average. The following illustrates the syntax of the Oracle AVG() function:

```
1 AVG([DISTINCT | ALL ] expression)
```

 For example, the following statement calculates the average of salary from employees table:

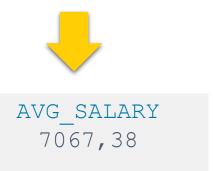
```
1 SELECT
2 ROUND(AVG( Salary ), 2) AVG_SALARY
3 FROM
4 EMPLOYEES;
5
```

AVG_SALARY 6461,83

Oracle AVG() with DISTINCT Clause

The following statement calculates the average DISTINCT salary:

```
1 SELECT
2 ROUND(AVG(DISTICT Salary),2) AVG_SALARY
3 FROM
4 EMPLOYEES;
5
```



COUNT()

 The Oracle COUNT() function is an aggregate function that returns the number of items in a group. The syntax of the COUNT() function is as follows:

```
1 COUNT( [ALL | DISTINCT | * ] expression)
```

 The following statement uses the COUNT(*) function to return the number of rows in the items table including NULL and duplicate values:

```
1 SELECT
2 COUNT(*)
3 FROM
4 EMPLOYEES;
```

28



- The Oracle MAX(), MIN() functions are aggregate functions that return the maximum and the minimum value of a set.
- The following illustrates the syntax of MAX() function:

```
1 MAX ( expression );
```

The following example returns the salary of the employee who receive maximum salary:

```
1 SELECT
2 MAX (Salary)
3 FROM
4 EMPLOYEES;
```



- The Oracle MIN() function is an <u>aggregate function</u> that returns the minimum value of a set.
- The syntax of the Oracle MIN() function is as follows:

```
1 MIN( expression );
```

The following example returns the salary of the employee who receive minimum salary:

```
1 SELECT
2 MIN(Salary)
3 FROM
4 EMPLOYEES;

MIN(Salary)
2100
```



- The Oracle SUM() function is an aggregate function that returns the sum of all or distinct values in a set of values.
- The following illustrates the syntax of the Oracle SUM() function:

```
1 SUM( [ALL | DISTINCT] expression)
```

The following statement returns the **sum** of salary that employees receive:

```
1 SELECT
2 SUM(Salary)
3 FROM
4 EMPLOYEES;
```

III Group by & Having clause



ORACLE GROUP BY

o The **GROUP BY** clause is used in a <u>SELECT</u> statement to group rows into a set of summary rows by values of columns or expressions. The <u>GROUP BY</u> clause returns one row per group. The following illustrates the syntax of the Oracle <u>GROUP BY</u> clause:

```
1 SELECT
2   column_list
3 FROM
4   T
5 GROUP BY c1,c2,c3;
```

o The GROUP BY clause groups rows by values in the grouping columns such as c1, c2 and c3. The GROUP BY clause must contain only aggregates or grouping columns.

ORACLE GROUP BY

 The GROUP BY clause appears after the FROM clause. In case WHERE clause is presented, the GROUP BY clause must be placed after the WHERE clause as shown in the following query:

```
1 SELECT
2   column_list
3 FROM
4   T
5 WHERE
6   condition
7 GROUP BY c1, c2, c3;
```

Oracle GROUP BY Examples

o Display manager ID and number of employees managed by the manager:

```
1 SELECT
2 MANAGER_ID, COUNT(*)
3 FROM
4 EMPLOYEES
5 GROUP BY
6 MANAGER_ID
```



| MANAGER_ID | COUNT(*) |
|------------|----------|
| (null) | 1 |
| 100 | 14 |
| 123 | 8 |
| 120 | 8 |
| 121 | 8 |
| 147 | 6 |
| | |

Oracle GROUP BY Examples

o Display the country ID and number of cities we have in the country:

```
COUNTRY_ID COUNT(*)
COUNTRY_ID COUNT(*)
US 4
SFROM
LOCATIONS
GROUP BY
CH 2
IT 2
```

o Display average salary of employees in each department who have commission percentage:

```
1 SELECT
2 DEPARTMENT_ID, AVG(SALARY)
3 FROM
4 EMPLOYEES WHERE COMMISSION_
5 PCT IS NOT NULL
6 GROUP BY
7 DEPARTMENT_ID
```

Oracle HAVING Clause

o The **HAVING** clause is an optional clause of the <u>SELECT</u> statement. It is used to filter groups of rows returned by the <u>GROUP BY</u> clause. This is why the <u>HAVING</u> clause is usually used with the <u>GROUP BY</u> clause.

The following illustrates the syntax of the Oracle HAVING clause:

```
1 SELECT
2 column_list
3 FROM
4 T
5 GROUP BY
6 c1
7 HAVING
8 group_condition;
```

- In this statement, the HAVING clause appears immediately after the GROUP BY clause.
- Note that the HAVING clause filters groups of rows while the WHERE clause filters rows. This is a main difference between the HAVING and WHERE clauses.

37

Oracle HAVING Clause example

Display job ID for jobs with average salary more than 10000.

```
SELECT
                                               JOB ID
                                                              AVG (SALARY)
                                                                  12008
   JOB ID, AVG(SALARY)
                                               AC MGR
FROM
                                               PU MAN
                                                                  11000
   EMPLOYEES
                                               AD VP
                                                                  17000
GROUP BY
                                                                  12008
                                               FI MGR
                                               SA MAN
                                                                  12200
   JOB ID
HAVING
                                              MK MAN
                                                                  13000
   AVG(SALARY)>10000
                                                                  24000
                                               AD PRES
```

Display departments in which more than five employees have commission percentage.

```
1 SELECT
2 DEPARTMENT_ID
3 FROM
4 EMPLOYEES
5 WHERE COMMISSION_PCT IS NOT NULL
6 GROUP BY
7 DEPARTMENT_ID
8 HAVING
COUNT (COMMISSION_PCT) > 5
```

Thank you! Data Science Academy