Oracle Schema Object Names and Qualifiers

Some schema objects are made up of parts that you can or must name, such as the columns in a table or view, index and table partitions and subpartitions, integrity constraints on a table, and objects that are stored within a package, including procedures and stored functions. This section provides:

- Rules for naming schema objects and schema object location qualifiers
- Guidelines for naming schema objects and qualifiers

Note:
Oracle uses system-generated names beginning with "SYS_" for implicitly generated schema objects and subobjects, and names beginning with "ORA_" for some Oracle-supplied objects. Oracle discourages you from using these prefixes in the names you explicitly provide to your schema objects and subobjects to avoid possible conflict in name resolution.

Schema Object Naming Rules

Every database object has a name. In a SQL statement, you represent the name of an object with a **quoted identifier** or a **nonquoted identifier**.

- A quoted identifier begins and ends with double quotation marks ("). If you name a schema object using a quoted identifier, then you must use the double quotation marks whenever you refer to that object.
- A nonquoted identifier is not surrounded by any punctuation.

You can use either quoted or nonquoted identifiers to name any database object. However, database names, global database names, and database link names are always case insensitive and are stored as uppercase. If you specify such names as quoted identifiers, then the quotation marks are silently ignored. Refer to CREATE USER for additional rules for naming users and passwords.

The following list of rules applies to both quoted and nonquoted identifiers unless otherwise indicated:

1. Names must be from 1 to 30 bytes long with these exceptions:
    - Names of databases are limited to 8 bytes.
    - Names of database links can be as long as 128 bytes.

   If an identifier includes multiple parts separated by periods, then each attribute can be up to 30 bytes long. Each period separator, as well as any surrounding double quotation marks, counts as one byte. For example, suppose you identify a column like this:

   "*schema*"."*table*"."*column*"

   The schema name can be 30 bytes, the table name can by 30 bytes, and the column name can be 30 bytes. Each of the quotation marks and periods is a single-byte character, so the total length of the identifier in this example can be up to 98 bytes.

2. Nonquoted identifiers cannot be Oracle Database reserved words. Quoted identifiers can be reserved words, although this is not recommended.

Depending on the Oracle product you plan to use to access a database object, names might be further restricted by other product-specific reserved words.

Note:

The reserved word ROWID is an exception to this rule. You cannot use the uppercase word ROWID, either quoted or nonquoted, as a column name. However, you can use the uppercase word as a quoted identifier that is not a column name, and you can use the word with one or more lowercase letters (for example, "Rowid" or "rowid") as any quoted identifier, including a column name.

3. The Oracle SQL language contains other words that have special meanings. These words include datatypes, schema names, function names, the dummy system table DUAL, and keywords (the uppercase words in SQL statements, such as DIMENSION, SEGMENT, ALLOCATE, DISABLE, and so forth). These words are not reserved. However, Oracle uses them internally in specific ways. Therefore, if you use these words as names for objects and object parts, then your SQL statements may be more difficult to read and may lead to unpredictable results.

   In particular, do not use words beginning with SYS_ or ORA_ as schema object names, and do not use the names of SQL built-in functions for the names of schema objects or user-defined functions.

4. You should use ASCII characters in database names, global database names, and database link names, because ASCII characters provide optimal compatibility across different platforms and operating systems.
5. Nonquoted identifiers must begin with an alphabetic character from your database character set. Quoted identifiers can begin with any character.
6. Nonquoted identifiers can contain only alphanumeric characters from your database character set and the underscore (_), dollar sign ($), and pound sign (#). Database links can also contain periods (.) and "at" signs (@). Oracle strongly discourages you from using $ and # in nonquoted identifiers.

   Quoted identifiers can contain any characters and punctuations marks as well as spaces. However, neither quoted nor nonquoted identifiers can contain double quotation marks or the null character (\0).

7. Within a namespace, no two objects can have the same name.

   The following schema objects share one namespace:

   - Tables
   - Views
   - Sequences
   - Private synonyms
   - Stand-alone procedures
   - Stand-alone stored functions
   - Packages
   - Materialized views
   - User-defined types

Each of the following schema objects has its own namespace:

- Indexes
- Constraints
- Clusters
- Database triggers
- Private database links
- Dimensions

Because tables and views are in the same namespace, a table and a view in the same schema cannot have the same name. However, tables and indexes are in different namespaces. Therefore, a table and an index in the same schema can have the same name.

Each schema in the database has its own namespaces for the objects it contains. This means, for example, that two tables in different schemas are in different namespaces and can have the same name.

Each of the following nonschema objects also has its own namespace:

- User roles
- Public synonyms
- Public database links
- Tablespaces
- Profiles
- Parameter files (PFILEs) and server parameter files (SPFILEs)

Because the objects in these namespaces are not contained in schemas, these namespaces span the entire database.

8. Nonquoted identifiers are not case sensitive. Oracle interprets them as uppercase. Quoted identifiers are case sensitive.

   By enclosing names in double quotation marks, you can give the following names to different objects in the same namespace:

   employees
   "employees"
   "Employees"
   "EMPLOYEES"

   Note that Oracle interprets the following names the same, so they cannot be used for different objects in the same namespace:

   employees
   EMPLOYEES
   "EMPLOYEES"

9. When Oracle stores or compares identifiers in uppercase, the uppercase form of each character in the identifiers is determined by applying the uppercasing rules of the database character set.

Language-specific rules determined by the session setting NLS_SORT are not considered. This behavior corresponds to applying the SQL function UPPER to the identifier rather than the function NLS_UPPER.

The database character set uppercasing rules can yield results that are incorrect when viewed as being in a certain natural language. For example, small letter sharp s ("ß"), used in German, does not have an uppercase form according to the database character set uppercasing rules. It is not modified when an identifier is converted into uppercase, while the expected uppercase form in German is the sequence of two characters capital letter S ("SS"). Similarly, the uppercase form of small letter i, according to the database character set uppercasing rules, is capital letter I. However, the expected uppercase form in Turkish and Azerbaijani is capital letter I with dot above.

The database character set uppercasing rules ensure that identifiers are interpreted the same in any linguistic configuration of a session. If you want an identifier to look correctly in a certain natural language, then you can quote it to preserve the lowercase form or you can use the linguistically correct uppercase form whenever you use that identifier.

10. Columns in the same table or view cannot have the same name. However, columns in different tables or views can have the same name.
11. Procedures or functions contained in the same package can have the same name, if their arguments are not of the same number and datatypes. Creating multiple procedures or functions with the same name in the same package with different arguments is called **overloading** the procedure or function.

Schema Object Naming Examples

The following examples are valid schema object names:

last_name
horse
hr.hire_date
"EVEN THIS & THAT!"
a_very_long_and_valid_name

All of these examples adhere to the rules listed in "Schema Object Naming Rules". The following example is not valid, because it exceeds 30 characters:

a_very_very_long_and_valid_name

Although column aliases, table aliases, usernames, and passwords are not objects or parts of objects, they must also follow these naming rules unless otherwise specified in the rules themselves.

Schema Object Naming Guidelines

Here are several helpful guidelines for naming objects and their parts:

- Use full, descriptive, pronounceable names (or well-known abbreviations).
- Use consistent naming rules.

- Use the same name to describe the same entity or attribute across tables.

When naming objects, balance the objective of keeping names short and easy to use with the objective of making names as descriptive as possible. When in doubt, choose the more descriptive name, because the objects in the database may be used by many people over a period of time. Your counterpart ten years from now may have difficulty understanding a table column with a name like pmdd instead of payment_due_date.

Using consistent naming rules helps users understand the part that each table plays in your application. One such rule might be to begin the names of all tables belonging to the FINANCE application with fin_.

Use the same names to describe the same things across tables. For example, the department number columns of the sample employees and departments tables are both named department_id.

Oracle Database Reserved Words

Words followed by an asterisk (*) are also ANSI reserved words.

Note:
In addition to the following reserved words, Oracle uses system- generated names beginning with "SYS_" for implicitly generated schema objects and subobjects. Oracle discourages you from using this prefix in the names you explicitly provide to your schema objects and subobjects to avoid possible conflict in name resolution.

ACCESS	EXCLUSIVE	MODIFY	SET *
ADD *	EXISTS	NOAUDIT	SHARE
ALL *	FILE	NOCOMPRESS	SIZE *
ALTER *	FLOAT *	NOT *	SMALLINT *
AND *	FOR *	NOWAIT	START
ANY *	FROM *	NULL *	SUCCESSFUL
AS *	GRANT *	NUMBER	SYNONYM
ASC *	GROUP *	OF *	SYSDATE
AUDIT	HAVING *	OFFLINE	TABLE *
BETWEEN *	IDENTIFIED	ON *	THEN *
BY *	IMMEDIATE *	ONLINE	TO *
CHAR *	IN *	OPTION *	TRIGGER
CHECK *	INCREMENT	OR *	UID
CLUSTER	INDEX	ORDER *	UNION *
COLUMN	INITIAL	PCTFREE	UNIQUE *
COMMENT	INSERT *	PRIOR *	UPDATE *
COMPRESS	INTEGER *	PRIVILEGES *	USER *
CONNECT *	INTERSECT *	PUBLIC *	VALIDATE
CREATE *	INTO *	RAW	VALUES *
CURRENT *	IS *	RENAME	VARCHAR *
DATE *	LEVEL *	RESOURCE	VARCHAR2
DECIMAL *	LIKE *	REVOKE *	VIEW *
DEFAULT *	LOCK	ROW	WHENEVER *
DELETE *	LONG	ROWID	WHERE
DESC *	MAXEXTENTS	ROWNUM	WITH *
DISTINCT *	MINUS	ROWS *	
DROP *	MLSLABEL	SELECT *	
ELSE *	MODE	SESSION *	

PL/SQL Reserved Words and Keywords

Both **reserved words** and **keywords** have special meaning in PL/SQL. The difference between reserved words and keywords is that you cannot use reserved words as identifiers. You can use keywords as as identifiers, but it is not recommended.

Table D-1 PL/SQL Reserved Words

Begins with:	**Reserved Words**
A	ALL, ALTER, AND, ANY, AS, ASC, AT
B	BEGIN, BETWEEN, BY
C	CASE, CHECK, CLUSTER, CLUSTERS, COLAUTH, COLUMNS, COMPRESS, CONNECT, CRASH, CREATE, CURRENT
D	DECLARE, DEFAULT, DELETE, DESC, DISTINCT, DROP
E	ELSE, END, EXCEPTION, EXCLUSIVE, EXISTS
F	FETCH, FOR, FROM
G	GOTO, GRANT, GROUP
H	HAVING
I	IDENTIFIED, IF, IN, INDEX, INDEXES, INSERT, INTERSECT, INTO, IS
L	LIKE, LOCK
M	MINUS, MODE
N	NOCOMPRESS, NOT, NOWAIT, NULL
O	OF, ON, OPTION, OR, ORDER, OVERLAPS
P	PRIOR, PROCEDURE, PUBLIC
R	RESOURCE, REVOKE
S	SELECT, SHARE, SIZE, SQL, START
T	TABAUTH, TABLE, THEN, TO
U	UNION, UNIQUE, UPDATE
V	VALUES, VIEW, VIEWS
W	WHEN, WHERE, WITH

Table D-2 PL/SQL Keywords

Begins with:	**Keywords**
A	A, ADD, AGENT, AGGREGATE, ARRAY, ATTRIBUTE, AUTHID, AVG
B	BFILE_BASE, BINARY, BLOB_BASE, BLOCK, BODY, BOTH, BOUND, BULK, BYTE
C	C, CALL, CALLING, CASCADE, CHAR, CHAR_BASE, CHARACTER, CHARSETFORM, CHARSETID, CHARSET, CLOB_BASE, CLOSE, COLLECT, COMMENT, COMMIT, COMMITTED, COMPILED, CONSTANT, CONSTRUCTOR, CONTEXT, CONTINUE, CONVERT, COUNT, CURSOR, CUSTOMDATUM

Begins with:	**Keywords**
D	DANGLING, DATA, DATE, DATE_BASE, DAY, DEFINE, DETERMINISTIC, DOUBLE, DURATION
E	ELEMENT, ELSIF, EMPTY, ESCAPE, EXCEPT, EXCEPTIONS, EXECUTE, EXIT, EXTERNAL
F	FINAL, FIXED, FLOAT, FORALL, FORCE, FUNCTION
G	GENERAL
H	HASH, HEAP, HIDDEN, HOUR
I	IMMEDIATE, INCLUDING, INDICATOR, INDICES, INFINITE, INSTANTIABLE, INT, INTERFACE, INTERVAL, INVALIDATE, ISOLATION
J	JAVA
L	LANGUAGE, LARGE, LEADING, LENGTH, LEVEL, LIBRARY, LIKE2, LIKE4, LIKEC, LIMIT, LIMITED, LOCAL, LONG, LOOP
M	MAP, MAX, MAXLEN, MEMBER, MERGE, MIN, MINUTE, MOD, MODIFY, MONTH, MULTISET
N	NAME, NAN, NATIONAL, NATIVE, NCHAR, NEW, NOCOPY, NUMBER_BASE
O	OBJECT, OCICOLL, OCIDATETIME, OCIDATE, OCIDURATION, OCIINTERVAL, OCILOBLOCATOR, OCINUMBER, OCIRAW, OCIREFCURSOR, OCIREF, OCIROWID, OCISTRING, OCITYPE, ONLY, OPAQUE, OPEN, OPERATOR, ORACLE, ORADATA, ORGANIZATION, ORLANY, ORLVARY, OTHERS, OUT, OVERRIDING
P	PACKAGE, PARALLEL_ENABLE, PARAMETER, PARAMETERS, PARTITION, PASCAL, PIPE, PIPELINED, PRAGMA, PRECISION, PRIVATE
R	RAISE, RANGE, RAW, READ, RECORD, REF, REFERENCE, RELIES_ON, REM, REMAINDER, RENAME, RESULT, RESULT_CACHE, RETURN, RETURNING, REVERSE, ROLLBACK, ROW
S	SAMPLE, SAVE, SAVEPOINT, SB1, SB2, SB4, SECOND, SEGMENT, SELF, SEPARATE, SEQUENCE, SERIALIZABLE, SET, SHORT, SIZE_T, SOME, SPARSE, SQLCODE, SQLDATA, SQLNAME, SQLSTATE, STANDARD, STATIC, STDDEV, STORED, STRING, STRUCT, STYLE, SUBMULTISET, SUBPARTITION, SUBSTITUTABLE, SUBTYPE, SUM, SYNONYM
T	TDO, THE, TIME, TIMESTAMP, TIMEZONE_ABBR, TIMEZONE_HOUR, TIMEZONE_MINUTE, TIMEZONE_REGION, TRAILING, TRANSACTION, TRANSACTIONAL, TRUSTED, TYPE
U	UB1, UB2, UB4, UNDER, UNSIGNED, UNTRUSTED, USE, USING
V	VALIST, VALUE, VARIABLE, VARIANCE, VARRAY, VARYING, VOID
W	WHILE, WORK, WRAPPED, WRITE
Y	YEAR
Z	ZONE

Names of Oracle Built-in Datatypes

VARCHAR2
NVARCHAR2
NUMBER
FLOAT
LONG
DATE
BINARY_FLOAT
BINARY_DOUBLE
TIMESTAMP
TIMESTAMP WITH TIME ZONE
TIMESTAMP WITH LOCAL TIME ZONE
INTERVAL YEAR TO MONTH
INTERVAL DAY TO SECOND
RAW
LONG RAW
ROWID
UROWID
CHAR
NCHAR
CLOB
NCLOB
BLOB
BFILE

Names of Oracle Functions

Numeric Functions

Numeric functions accept numeric input and return numeric values. Most numeric functions that return NUMBER values that are accurate to 38 decimal digits. The transcendental functions COS, COSH, EXP, LN, LOG, SIN, SINH, SQRT, TAN, and TANH are accurate to 36 decimal digits. The transcendental functions ACOS, ASIN, ATAN, and ATAN2 are accurate to 30 decimal digits. The numeric functions are:

ABS	COS	NANVL	SQRT
ACOS	COSH	POWER	TAN
ASIN	EXP	REMAINDER	TANH
ATAN	FLOOR	ROUND (number)	TRUNC (number)
ATAN2	LN	SIGN	WIDTH_BUCKET
BITAND	LOG	SIN	
CEIL	MOD	SINH	

Character Functions Returning Character Values

Character functions that return character values return values of the following datatypes unless otherwise documented:

- If the input argument is CHAR or VARCHAR2, then the value returned is VARCHAR2.
- If the input argument is NCHAR or NVARCHAR2, then the value returned is NVARCHAR2.

The length of the value returned by the function is limited by the maximum length of the datatype returned.

- For functions that return CHAR or VARCHAR2, if the length of the return value exceeds the limit, then Oracle Database truncates it and returns the result without an error message.
- For functions that return CLOB values, if the length of the return values exceeds the limit, then Oracle raises an error and returns no data.

The character functions that return character values are:

CHR	NLS_INITCAP	REPLACE	TREAT
CONCAT	NLS_LOWER	RPAD	TRIM
INITCAP	NLSSORT	RTRIM	UPPER
LOWER	NLS_UPPER	SOUNDEX	
LPAD	REGEXP_REPLACE	SUBSTR	
LTRIM	REGEXP_SUBSTR	TRANSLATE	

NLS Character Functions

The NLS character functions return information about the character set. The NLS character functions are:

NLS_CHARSET_DECL_LEN

NLS_CHARSET_ID
NLS_CHARSET_NAME

Character Functions Returning Number Values

Character functions that return number values can take as their argument any character datatype.

The character functions that return number values are:

ASCII
INSTR
LENGTH
REGEXP_INSTR

Datetime Functions

Datetime functions operate on date (DATE), timestamp (TIMESTAMP, TIMESTAMP WITH TIME ZONE, and TIMESTAMP WITH LOCAL TIME ZONE), and interval (INTERVAL DAY TO SECOND, INTERVAL YEAR TO MONTH) values.

Some of the datetime functions were designed for the Oracle DATE datatype (ADD_MONTHS, CURRENT_DATE, LAST_DAY, NEW_TIME, and NEXT_DAY). If you provide a timestamp value as their argument, then Oracle Database internally converts the input type to a DATE value and returns a DATE value. The exceptions are the MONTHS_BETWEEN function, which returns a number, and the ROUND and TRUNC functions, which do not accept timestamp or interval values at all.

The remaining datetime functions were designed to accept any of the three types of data (date, timestamp, and interval) and to return a value of one of these types.

All of the datetime functions that return current system datetime information, such as SYSDATE, SYSTIMESTAMP, CURRENT_TIMESTAMP, and so forth, are evaluated once for each SQL statement, regardless how many times they are referenced in that statement.

The datetime functions are:

ADD_MONTHS	NEW_TIME	TO_CHAR (datetime)
CURRENT_DATE	NEXT_DAY	TO_TIMESTAMP
CURRENT_TIMESTAMP	NUMTODSINTERVAL	TO_TIMESTAMP_TZ
DBTIMEZONE	NUMTOYMINTERVAL	TO_DSINTERVAL
EXTRACT (datetime)	ROUND (date)	TO_YMINTERVAL
FROM_TZ	SESSIONTIMEZONE	TRUNC (date)
LAST_DAY	SYS_EXTRACT_UTC	TZ_OFFSET
LOCALTIMESTAMP	SYSDATE	
MONTHS_BETWEEN	SYSTIMESTAMP	

General Comparison Functions

The general comparison functions determine the greatest and or least value from a set of values. The general comparison functions are:

GREATEST
LEAST

Conversion Functions

Conversion functions convert a value from one datatype to another. Generally, the form of the function names follows the convention *datatype* TO *datatype*. The first datatype is the input datatype. The second datatype is the output datatype. The SQL conversion functions are:

ASCIISTR	ROWIDTONCHAR	TO_NCHAR (character)
BIN_TO_NUM	SCN_TO_TIMESTAMP	TO_NCHAR (datetime)
CAST	TIMESTAMP_TO_SCN	TO_NCHAR (number)
CHARTOROWID	TO_BINARY_DOUBLE	TO_NCLOB
COMPOSE	TO_BINARY_FLOAT	TO_NUMBER
CONVERT	TO_CHAR (character)	TO_DSINTERVAL
DECOMPOSE	TO_CHAR (datetime)	TO_SINGLE_BYTE
HEXTORAW	TO_CHAR (number)	TO_TIMESTAMP
NUMTODSINTERVAL	TO_CLOB	TO_TIMESTAMP_TZ
NUMTOYMINTERVAL	TO_DATE	TO_YMINTERVAL
RAWTOHEX	TO_DSINTERVAL	TO_YMINTERVAL
RAWTONHEX	TO_LOB	TRANSLATE ... USING
ROWIDTOCHAR	TO_MULTI_BYTE	UNISTR

Large Object Functions

The large object functions operate on LOBs. The large object functions are:

BFILENAME
EMPTY_BLOB, EMPTY_CLOB

Collection Functions
The collection functions operate on nested tables and varrays. The SQL collection functions are:

CARDINALITY
COLLECT
POWERMULTISET
POWERMULTISET_BY_CARDINALITY
SET

Hierarchical Function

The hierarchical function applies hierarchical path information to a result set.

SYS_CONNECT_BY_PATH

Data Mining Functions

The data mining functions operate on models that have been built using the DBMS_DATA_MINING package or the Oracle Data Mining Java API. The SQL data mining functions are:

CLUSTER_ID	PREDICTION
CLUSTER_PROBABILITY	PREDICTION_BOUNDS
CLUSTER_SET	PREDICTION_COST
FEATURE_ID	PREDICTION_DETAILS
FEATURE_SET	PREDICTION_PROBABILITY
FEATURE_VALUE	PREDICTION_SET

XML Functions

APPENDCHILDXML	SYS_XMLGEN	XMLFOREST
DELETEXML	UPDATEXML	XMLPARSE
DEPTH	XMLAGG	XMLPATCH
EXTRACT (XML)	XMLCAST	XMLPI
EXISTSNODE	XMLCDATA	XMLQUERY
EXTRACTVALUE	XMLCOLATTVAL	XMLROOT
INSERTCHILDXML	XMLCOMMENT	XMLSEQUENCE
INSERTXMLBEFORE	XMLCONCAT	XMLSERIALIZE
PATH	XMLDIFF	XMLTABLE
SYS_DBURIGEN	XMLELEMENT	XMLTRANSFORM
SYS_XMLAGG	XMLEXISTS	

Encoding and Decoding Functions
The encoding and decoding functions let you inspect and decode data in the database.

DECODE	ORA_HASH
DUMP	VSIZE

NULL-Related Functions
The NULL-related functions facilitate null handling. The NULL-related functions are:

COALESCE	NVL
LNNVL	NVL2
NULLIF	

Environment and Identifier Functions

The environment and identifier functions provide information about the instance and session. These functions are:

SYS_CONTEXT	UID
SYS_GUID	USER
SYS_TYPEID	USERENV

Aggregate Functions

Aggregate functions return a single result row based on groups of rows, rather than on single rows. Aggregate functions can appear in select lists and in ORDER BY and HAVING clauses. They are commonly used with the GROUP BY clause in a SELECT statement, where Oracle Database divides the rows of a queried table or view into groups. In a query containing a GROUP BY clause, the elements of the select list can be aggregate functions, GROUP BY expressions, constants, or expressions involving one of these. Oracle applies the aggregate functions to each group of rows and returns a single result row for each group.

If you omit the GROUP BY clause, then Oracle applies aggregate functions in the select list to all the rows in the queried table or view. You use aggregate functions in the HAVING clause to eliminate groups from the output based on the results of the aggregate functions, rather than on the values of the individual rows of the queried table or view.

Many (but not all) aggregate functions that take a single argument accept these clauses:

- DISTINCT causes an aggregate function to consider only distinct values of the argument expression.
- ALL causes an aggregate function to consider all values, including all duplicates.

For example, the DISTINCT average of 1, 1, 1, and 3 is 2. The ALL average is 1.5. If you specify neither, then the default is ALL.

All aggregate functions except COUNT(*), GROUPING, and GROUPING_ID ignore nulls. You can use the NVL function in the argument to an aggregate function to substitute a value for a null. COUNT and REGR_COUNT never return null, but return either a number or zero. For all the remaining aggregate functions, if the data set contains no rows, or contains only rows with nulls as arguments to the aggregate function, then the function returns null.

The aggregate functions MIN, MAX, SUM, AVG, COUNT, VARIANCE, and STDDEV, when followed by the KEEP keyword, can be used in conjunction with the FIRST or LAST function to operate on a set of values from a set of rows that rank as the FIRST or LAST with respect to a given sorting specification.

You can nest aggregate functions. For example, the following example calculates the average of the maximum salaries of all the departments in the sample schema hr:

```
SELECT AVG(MAX(salary)) FROM employees GROUP BY department_id;
```

```
AVG(MAX(SALARY))
----------------
           10925
```

This calculation evaluates the inner aggregate (MAX(salary)) for each group defined by the GROUP BY clause (department_id), and aggregates the results again.

The aggregate functions are:

AVG	RANK
COLLECT	REGR_ (Linear Regression) Functions
CORR	STATS_BINOMIAL_TEST
CORR_*	STATS_CROSSTAB
COUNT	STATS_F_TEST
COVAR_POP	STATS_KS_TEST
COVAR_SAMP	STATS_MODE
CUME_DIST	STATS_MW_TEST
DENSE_RANK	STATS_ONE_WAY_ANOVA
FIRST	STATS_T_TEST_*
GROUP_ID	STATS_WSR_TEST
GROUPING	STDDEV
GROUPING_ID	STDDEV_POP
LAST	STDDEV_SAMP
MAX	SUM
MEDIAN	SYS_XMLAGG
MIN	VAR_POP
PERCENTILE_CONT	VAR_SAMP
PERCENTILE_DISC	VARIANCE
PERCENT_RANK	XMLAGG