# C993 Notes

## Creating Groups of Data

<https://docs.oracle.com/cd/B28359_01/server.111/b28286/functions064.htm#SQLRF00647>

<https://www.oracletutorial.com/oracle-basics/oracle-group-by/>

<https://blog.jooq.org/2016/12/09/a-beginners-guide-to-the-true-order-of-sql-operations/>



Grouping distinguishes superaggregrate rows from regular grouped rows.

COUNT(\*) will return 0 if no rows returned. Not true for other aggregate functions.

Group functions will return summary information per group. GROUP BY is often used with aggregate functions such a AVG(), COUNT(), MAX(), MIN(), SUM().

Group functions can only be nested to two levels.

Nested group functions require a group by clause.

**aggregations**: This is important to understand. No matter where you put your aggregate function syntactically (i.e. in the SELECT clause, or in the ORDER BY clause), this here is the step where aggregate functions are calculated. Right after GROUP BY. (remember: logically. Clever databases may have calculated them before, actually). This explains why you cannot put an aggregate function in the WHERE clause, because its value cannot be accessed yet. The WHERE clause logically happens before the aggregation step. Aggregate functions can access columns that you have put in “this list” for each group, above. After aggregation, “this list” will disappear and no longer be available. If you don’t have a GROUP BY clause, there will just be one big group without any key, containing all the rows.

-- Wrong

SELECT first\_name, count(\*)

FROM customer

WHERE count(\*) > 1

GROUP BY first\_name

-- Correct

SELECT first\_name, count(\*)

FROM customer

GROUP BY first\_name

HAVING count(\*) > 1

-- Correct

SELECT first\_name, count(\*)

FROM customer

GROUP BY first\_name

ORDER BY count(\*) DESC

-- Wrong

SELECT first\_name, last\_name, count(\*)

FROM customer

GROUP BY first\_name

-- Correct

SELECT first\_name, MAX(last\_name), count(\*)

FROM customer

GROUP BY first\_name

-- Wrong

SELECT first\_name || ' ' || last\_name, count(\*)

FROM customer

GROUP BY first\_name

-- Correct

SELECT first\_name || ' ' || MAX(last\_name), count(\*)

FROM customer

GROUP BY first\_name

-- Correct

SELECT MAX(first\_name || ' ' || last\_name), count(\*)

FROM customer

GROUP BY first\_name

## Creating and using Temporary Tables

<https://docs.oracle.com/cd/B28359_01/server.111/b28318/schema.htm#CNCPT1138>

<https://oracle-base.com/articles/misc/temporary-tables>

<https://oracle-base.com/articles/18c/private-temporary-tables-18c>

### Global Temporary Table

* Can be transaction specific or session specific. If transacation specific the table exists for the duration of the transaction. If session it lasts for the duration of that session.
* The data structure is visible to ALL current sessions connected to the DB. But the data is private to each session.
* Truncate on a GTT is scope specific to the session that issued it.
* Data is dropped on session termination.
* Indexes created on temporary tables are also temporary.
* Syntax:
  + CREATE GLOBAL TEMPORARY TABLE table\_name (column datatype, …);
  + ON COMMIT you can DELETE ROWS (transaction specific table) or PRESERVE ROWS (session specific table).
  + CREATE GLOBAL TEMPORARY TABLE admin\_work\_area
  + (startdate DATE,
  + enddate DATE,
  + class CHAR(20))
  + ON COMMIT DELETE ROWS;
* Segments are allocated on the first insert (or create table as select). If a select, update, or delete are performed before an insert then the table appears empty.
* DDL statements can only be performed if no session is bound to the table.
* Sessions can be unbound by truncate, session termination, or performing a commit or rollback.

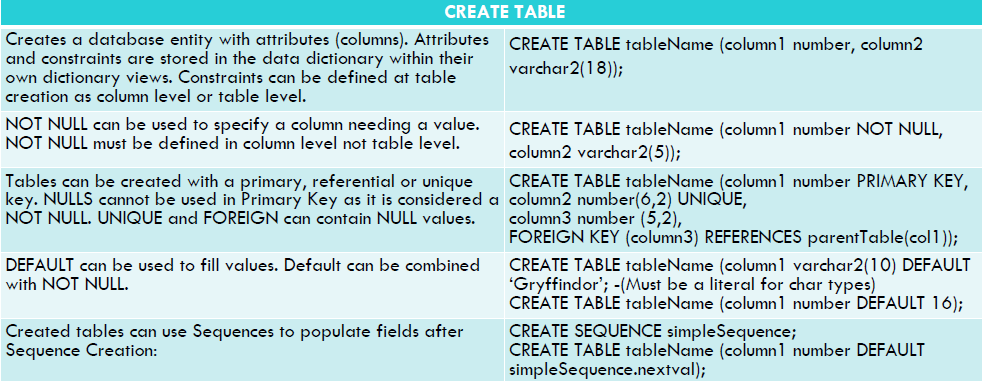
### Private temporary Tables

* Syntax:
  + CREATE PRIVATE TEMPORARY TABLE ORA$PTT\_table\_name (column datatype, ..);
* Data is visible only to the session which created it.
* PTT are placed in memory.

## Creating tables

<https://docs.oracle.com/cd/B28359_01/server.111/b28310/tables003.htm#ADMIN01503>

* To create a table in your schema you must have the CREATE TABLE system privilege. To create in someone else’s schema you must have the CREATE ANY TABLE system privilege. Also you must have a quota for tablespace or the UNLIMITED TABLESPACE privilege.
* Syntax:
  + CREATE TABLE table\_name (column datatype,…);



* A virtual column is a table column whose values are calculated automatically using other column values, or another deterministic expression.
* When creating a table as a select only the NOT NULL constraint may transfer.
* Syntax:
  + CREATE TABLE tableName2 AS SELECT \* FROM tableName1;

## Describing and Working with Columns and Data Types

* CHAR – defaults to 1 byte if unspecified. Max 2000 bytes. Will pad with blanks if less than the max length. If too large the interpreter will return an error.
* VARCHAR / VARCHAR2 – Length MUST be specified. 1 to 4000 bytes allowed. Can be specified in bytes or characters. Values are not padded.
  + VARCHAR2(20 BYTE) and SUBSTRB(<string>, 1, 20) use byte semantics. VARCHAR2(10 CHAR) and SUBSTR(<string>, 1, 10) use character semantics.
* NCHAR / NVARCHAR2 – Store Unicode character data. NCHAR length defaults to 1, NVARCHAR2 MUST be specified. NCHAR max is 2000bytes, NVARCHAR2 max is 4000 bytes.
* LONG is not recommended/ CLOB, NCLOB are used instead.
* NUMBER (precision, scale) – Not necessary to specify precision and scale. Max of 38 digits of precision. You can specify scale without precision by using a \* ex NUMBER (\*, scale)
* NUMERIC same as NUMBER
* DATE – Stores date info down to the minutes and seconds. Default format is DD-MON-YY
* TIMESTAMP – Date that goes down to fraction seconds (FF). Precision of fractional seconds defaults to 6

## Differentiating system privileges from object privileges | Distinguishing between granting privileges and roles | Granting privileges on tables

<https://docs.oracle.com/database/121/TTSQL/privileges.htm#TTSQL338>

<https://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_9013.htm>

<https://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_9020.htm>

* System privilege is the right to perform a particular action or to perform an action on any object of a particular type. Basically performing a database task.
  + GRANT \_\_\_ TO user;  
    REVOKE \_\_\_ FROM user;
  + ALTER ANY  
    CREATE ANY [any is for any user]  
    CREATE [w/o any the privilege only applies within the scope of the user]  
    CREATE SEQUENCE  
    DROP ANY  
    … INDEX, ROLE, SEQUENCE, TABLE, VIEW  
    CREATE SESSION  
    INSERT ANY  
    SELECT ANY
* Object privileges perform tasks on a specific object
  + GRANT \_\_\_\_ ON object TO user;
  + REVOKE \_\_\_ ON object TO user;
  + ALL  
    DELETE  
    INDEX  
    INSERT  
    REFERENCES  
    SELECT  
    UPDATE
* CREATE SESSION system privilege needed to connect to the database.
* If a permission is granted to PUBLIC that encompasses all users.
* Objects include tables, view, materialized views, synonyms, indexes, sequences, cache groups, replication schemes, and PL/SQL functions, procedures and packages.
* Only the instance administrator or a user with ADMIN privilege can grant and revoke system privileges.
* Object privileges give you the right to perform a particular action on an object or the access to another user’s object. An object owner has all object privileges for that object and those privileges cannot be revoked.
* ADMIN confers all other privileges.
* A role is a named group of related privileges that can be granted to the user. This method makes it easier to revoke and maintain privileges.
* A user can have access to multiple roles, and multiple users can be assigned to the same role.
* If a user is granted a role that contains a privilege, and also has that privilege. Removing the role or the privilege will still leave them with the privilege through either the role or privilege (confusing, but basically you would have to remove both).
* WITH ADMIN OPTION is only used with system privileges.
* WITH ADMIN gives you permission to grant something to other people.
* Revoke doesn’t cascade for DDL privileges but does for DML privileges.
* GRANT ALL PRIVILEGES: privileges is not optional for system privileges but is for object.
* WITH GRANT OPTION is used for object privileges only.
* OBJECT privileges cascade, system privileges do not.

## Dropping columns and setting column UNUSED

<https://docs.oracle.com/cd/B28359_01/server.111/b28310/tables006.htm#ADMIN01504>

<https://docs.oracle.com/cd/B28359_01/server.111/b28286/statements_3001.htm#SQLRF01001>

* Dropping Columns
  + You can drop columns that are no longer needed from a table
  + You cannot drop all columns from a table or drop columns from a table owned by SYS
  + DROP COLUMN is issued from the ALTER TABLE command:
    - ALTER TABLE hr.admin\_emp DROP COLUMN sal;
  + If you drop or mark unused a column defined as an INCLUDING column then the column stored immediately before this column will become the new INCLUDING column
* Setting Columns Unused
  + ALTER TABLE table SET UNUSED
  + Unused columns are treated as if they were DROPPED!
  + Setting a column unused is faster than dropping.
  + Marking one or more columns as unused does not actually remove the target column data or restore disk space occupied by these column.
  + An unused column is not displayed in queries or data dictionaries. The name is removed so another column may reuse that name.
  + You can later remove unused columns. ALTER TABLE table DROP UNUSED COLUMNS
  + An optional CHECKPOINT clause is allowed to reduce undo logs.

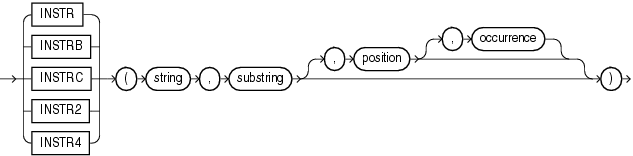
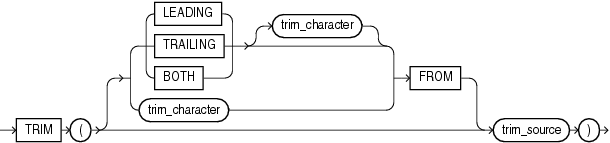
## Explaining the relationship between a database and SQL

* SQL is a language to create databases and objects within them.
  + Store data in databases
  + Change and analyze data
  + Get data back out
* DDL – Data definition Language
  + CREATE, ALTER, DROP, RENAME, TRUNCATE, GRANT, REVOKE, FLASHBACK, PURGE, COMMENT
* DML -Data Manipulation Language
  + SELECT, INSERT, UPDATE, DELETE, MERGE
* TCL – Transaction Control Language
  + COMMIT, ROLLBACK, SAVEPOINT

## Managing Sequences

* CREATE SEQUENCE sequence   
  Options : [INCREMENT BY | START WITH] – defaults to increment of 1. The start with defaults to the minimum value of incrementation  
  [MAXVALUE integer | NOMAXVALUE] – Default is nomaxvalue  
  [MINVALUE integer | NOMINVALUE] – default is nominvalue  
  [CYCLE | NOCYCLE] – default is no cycle  
  [ORDER | NOORDER]
* You have to call sequence.NEXTVAL in a session before you can call the .CURRVAL
* If .NEXTVAL was used in a script and the script failed the sequence still increments
* Anytime .NEXTVAL is called the value is taken and cannot be released without dropping the sequence.
* If you want to restart the sequence at a different number, you will need to drop and recreate a new one.

## Manipulating strings with character functions in SQL SELECT and WHERE Clauses

* Case-conversion
  + LCASE or LOWER( CharacterExpression)
  + UCASE or UPPER( CharacterExpression)
  + INITCAP(expression)
    - Converts the first letter of each word to upper case. The rest will be in lower case.
* Character Manipulation
  + CONCAT( expression1, expression2 )
    - Can only take two arguments. Piping expression1 || expression2 is the same, but can add more arguments.
  + SUBSTR(char, position, substring\_length[optional])
    - Returns a portion of char. If position is 0 then treated as 1. Position starts at 1.
    - If position is negative then it counts backwards from the end of char
    - If length is omitted, all characters will be returned until the end of char.
  + LENGTH
    - Returns the number of characters in a string
  + INSTR
    - 
    - Returns the numeric position of a named string. INSTRB uses bytes, INSTRC uses Unicode characters, INSTR2 uses UCS2 code points, INSTR4 uses UCS4 code points
    - Position indicates the starting position for the search
    - Occurrence is indicated which occurrence is being looked for. Must be positive.
  + LPAD
    - 
    - LPAD returns expression 1 padded to length n with the characters in expr2. Expr2 is blank if unspecified.
  + RPAD
    - Same as lpad but right padded
  + TRIM
  + 
    - Enables you to trim leading or trailing characters or both. Default is both.
    - Can specify a trim character. Defaulted to blank spaces
    - Returns the string as a VARCHAR2 if trim source is a character or LOB if a LOB datatype.
      * SELECT employee\_id,
      * TO\_CHAR(TRIM(LEADING 0 FROM hire\_date))
      * FROM employees
      * WHERE department\_id = 60;
  + REPLACE
    - 
    - REPLACE searches the char expression for a character string and replaces it with the specified replacement string. If replacement string is omitted then all instances of the search string are removed.

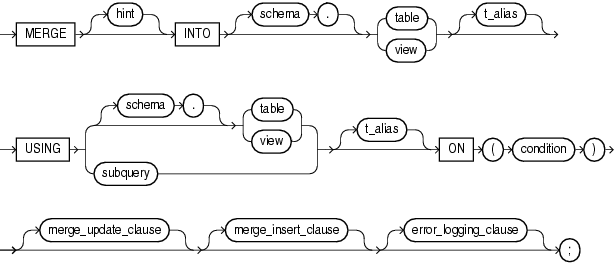
## Matching the SELECT statements

* Set operators require the same number of columns in each select statement
* UNION combines row sets and eliminates duplicate rows
* UNION ALL combines row sets and does not delimitate duplicate rows
* See set operators below for INTERSECT and MINUS

## Perform Insert, Update and Delete operations | Performing multi table Inserts

* DML Commands
* <https://docs.oracle.com/database/121/SQLRF/statements_9015.htm#SQLRF01604>
* INSERT INTO tbl\_name (tbl\_columns) VALUES (tbl\_values);
  + Table columns after table name is optional, but you must list a value for each column in the order the columns are in the schema.
* Multi table insert uses INSER ALL before the INTO statements. Ends with a subquery that specifies the rows that are inserted into the table.
* For INSERT ALL the subquery can refer to any table, view, or materialized view, including the target table of the INSERT statement. If the subquery selects no rows, then the database inserts no rows into the table.
* <https://docs.oracle.com/database/121/SQLRF/statements_10008.htm#SQLRF01708>
* UPDATE table\_name SET column1 = value,… WHERE x
* Update will update columns for all rows that satisfy the where condition
* If WHERE is excluded all rows are updated
* <https://docs.oracle.com/database/121/SQLRF/statements_8005.htm#SQLRF01505>
* DELETE statement removes rows
* DELETE FROM table\_name WHERE x
* FROM and WHERE are optional
* If WHERE is excluded or invalid all rows are deleted.
* DELETE vs TRUNCATE vs DROP
  + DROP and TRUNCATE are DDL. DELETE is DML
  + Truncate removes all rows, no rollback or flashback allowed.
  + Faster than DELETE, truncate does not fire DELETE triggers or generate rollback info
  + TRUNCATE can not be used to selective remove rows
  + TRUNCATE reallocates table space
  + DROP removes the entire table, all rows, indexes, and grants.
  + On drop the table is moved to the recycle bin. A drop does not reallocate tablespace.
  + DELETE allows rollback if no commit or ddl has occurred. Flashback may be allowed.

## Performing Merge statements

* <https://docs.oracle.com/database/121/SQLRF/statements_9017.htm#SQLRF01606>
* A merge can combine the functionality of insert update and delete
* 
* A clause must be specified (update or insert)
* ON clause can be complex comparisons or primary keys.
* UPDATE can not work on anything in the ON clause
* DELETE must be part of WHEN MATCHED
* Merge syntax:
* MERGE INTO table\_name

USING table | subquery | view

ON (join\_condition)

WHEN MATCHED THEN UPDATE SET column1 = expression

DELETE WHERE condition

WHEN NOT MATCHED THEN INSERT (column1, column2)

VALUES (expression1, expression2)

WHERE condition

## Performing arithmetic with date data

* The Oracle database stores dates in an internal numeric format. The default display is DD-MON-RR
* Date + number = date: add a number of days to a date
* Date - number = date: subtract a number of days from a date
* Date - date = number of days: subtracts one date from another and returns number of
* days
* Date + number/24 = date: adds a number of hours to a date.
* DATE + DATE IS NOT ALLOWED!
* MONTHS\_BETWEEN(date1, date2)
  + Returns the month between the 2 dates. If date1 is before date2 the result is negative.
* ADD\_MONTHS(date, n)
  + Adds ‘n’ months to date. There is no SUBTRACT\_MONTHS function. Just add a negative value.
* LAST\_DAY(date)
  + Returns the last day of the month
* ROUND(date, fmt) / TRUNC(date, fmt)
  + <https://docs.oracle.com/cd/B19306_01/server.102/b14200/functions230.htm#i1002084>
  + Fmt is optional default is DD for days.
  + ROUND will round the date forward to midnight
  + TRUNC will cut the day to midnight of the current day
* NEXT\_DAY(date, day)
  + Requires a valid day string and can also use numbers (enumerated with 1 as Sunday)
* NUMTOYMINTERVAL (n, interval\_unit)
  + Translates a number to year or month
* NUMTODSINTERVAL(n, interval\_unit)
  + Translates a number to day or second.

## Understanding and Using Cartesian Products

* Also known as a CROSS JOIN
* Combines one row in a table with every row in another table.
* Multiplies the total amount of rows in a table by the total amount of rows in another table

## Understanding implicit and explicit data type conversion

* Oracle can convert VARCHAR2 or CHAR to NUMBER if it is a valid number
* Can convert VARCHAR2 or CHAR to DATE if it is a valid date
* Can convert NUMBER to VARCHAR2 or CHAR using CONCAT function
* Can convert DATE to VARCHAR2 or CHAR using CONCAT function

## Update and delete rows using correlated subqueries

* An UPDATE statement can have a correlated subquery in these places.
  + In the SET clause
  + In the WHERE clause
* UPDATE INVOICES INV   
   SET TERMS\_OF\_DISCOUNT = '10 PCT'   
  WHERE TOTAL\_PRICE = (SELECT MAX(TOTAL\_PRICE)  
   FROM INVOICES  
   WHERE TO\_CHAR(INVOICE\_DATE, 'RRRR-Q') =   
   TO\_CHAR(INV.INVOICE\_DATE, 'RRRR-Q'));
* The DELETE statement can be used with a correlated subquery in the WHERE clause to determine which rows to delete from a given table.  
    
  DELETE FROM SHIP\_CABINS S1   
  WHERE S1.BALCONY\_SQ\_FT =   
   (SELECT MIN(BALCONY\_SQ\_FT)  
   FROM SHIP\_CABINS S2   
   WHERE S1.ROOM\_TYPE = S2.ROOM\_TYPE   
   AND S1.ROOM\_STYLE = S2.ROOM\_STYLE);

## Using Column aliases

* SELECT column\_name column\_alias | optional AS between column name and alias.
* If enclosed in double quotes the alia can include spaces. Must be reference with double quotes
* You can use the column alias in an ORDER BY statement
  + ORDER OF PROCESSING
    - FROM, WHERE, GROUP BY, HAVING, SELECT, ORDER BY, FETCH
* The alias can not be used before it is created in the SELECT statement in the order of processing.

## Using Multiple Row Subqueries

<https://docs.oracle.com/database/121/SQLRF/statements_10002.htm#SQLRF01702>

<https://docs.oracle.com/database/121/SQLRF/queries001.htm#SQLRF52327>

* Multiple row subqueries return multiple rows
* Multiple row subqueries can be used in a WHERE clause and the INTO portion of an INSERT statement. A multiple operater must be used suchas IN, ANY, or ALL
* When used in the INTO portion of an insert statement, all rows returned by the multiple-row subquery are inserted into the specified table.

## Using OUTER joins

* Includes items matched and unmatched from rows of two or more tables. Views data that doesn’t normally meet a join condition.
* OUTER keyword is optional (LEFT OUTER JOIN is the same as LEFT JOIN)
* LEFT JOIN: Returns all rows in the left table, and matching rows from the right
* RIGHT JOIN: Returns all from the right table and matched data from the left.
* FULL JOIN: Includes all unmatched and matched data from both sides.

## Using Self-joins

* Join a table to itself.
* SELECT a.position\_id, a.position, b.position boss

FROM positions a

LEFT OUTER JOIN positions b ON a.reports\_to = b.position\_id

ORDER BY a.position\_id;

## Using The INTERSECT operator

<https://docs.oracle.com/cd/B19306_01/server.102/b14200/queries004.htm>

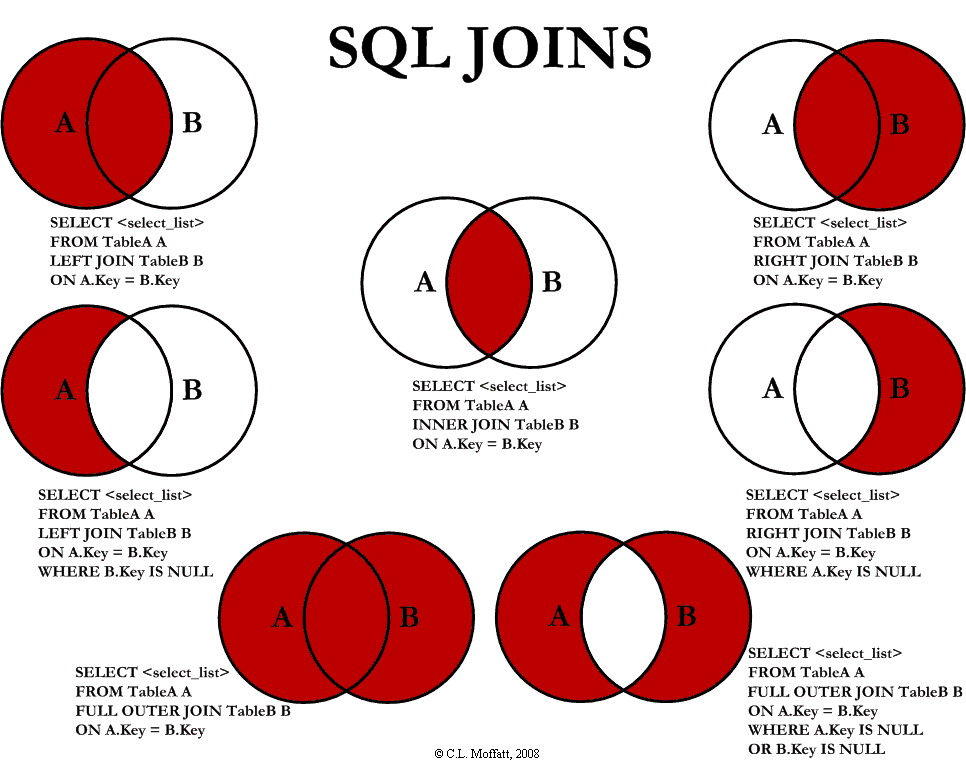
* Returns only the rows in both queries.
* SELECT product\_id FROM inventories
* INTERSECT
* SELECT product\_id FROM order\_items;

## Using The MINUS operator

* The order of the tables matters! It determines the output. Which table is being subtracted from which.
* When combining results with the MINUS operator you are left with the unique rows from the first query that were not in the second.
* SELECT product\_id FROM inventories
* MINUS
* SELECT product\_id FROM order\_items;

## Using Various Types of Joins

* See above for cross joins, left, right outer joins, self join, full join



## Using concatenation operator, literal character strings, alternative quote

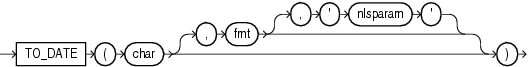
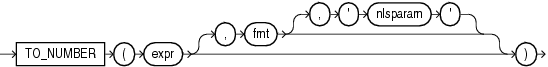
## operator, and the DISTINCT keyword

* CONCAT command only allows the concatenation of 2 strings. But using pipes you can concat as many strings as you want
* SELECT ‘Hello, ‘ || ‘World!’ FROM DUAL;
* Literal is a character string in quotations. Literals can be used for alias. And if using a literal reserved words can be used.
* You can use brackets and the letter q as an alternative quote operator
  + WHERE column\_name=q’{Spencer’s Gifts}’;
  + Doesn’t matter which brackets you use. () also allowed. This string could be created using double apostrophe’s e.g. column\_name = ‘Spencer’’s Gifts’
* The DISTINCT operator eliminates duplicates
* DISTINCT must be declared on the 1st Column or will be invalid.
* If multiple columns, DISTINCT will eliminate duplicates of the pairs.
* It is not considered an aggregate column in GROUP BY

## Using data dictionary views

* The data dictionary is owned by SYS
* It includes metadata for all of the database including objects.
* 5 different prefixes for most views
  + USER – Lists objects and views owned only by the current user
  + ALL – lists objects and views owned by and permitted to the current user
  + DBA – Lists all objects and views inside the database
  + V\_$ - Dynamic performance view. Local db instance
  + GV\_$ - global dynamic performance views
* Comments can be used on tables and columns to add clarification or instruction
* A comment can’t be dropped. It is common to just overwrite with a blank string.
* Syntax: COMMENT ON COLUMN employees.job\_id IS ‘this is a comment’;

## Using the TO\_CHAR, TO\_NUMBER, and TO\_DATE conversion functions

* TO\_CHAR(date, ‘format model’)
* <https://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_commands_1029.htm#OLADM780>
* Above link has all date formats
* 
* TO\_DATE converts char, varchar2, nchar, nvarchar2 to DATE type.
* 
* TO\_NUMBER converts the expression to a NUMBER data type. Expression can be a char, varchar2, nchar, nvarchar2, binary\_float, or binary\_double
* Binary floats or doubles cannot specify a format model.

## Working with CURRENT\_DATE, CURRENT\_TIMESTAMP,and

## LOCALTIMESTAMP

* <https://docs.oracle.com/database/121/SQLRF/functions051.htm#SQLRF00628>
* CURRENT\_DATE returns the current date in the session time zone (vs SYSDATE which would return the server time).
* ALTER SESSION SET TIME\_ZONE = '-5:0';
* ALTER SESSION SET NLS\_DATE\_FORMAT = 'DD-MON-YYYY HH24:MI:SS';
* SELECT SESSIONTIMEZONE, CURRENT\_DATE FROM DUAL;
* SESSIONTIMEZONE CURRENT\_DATE
* --------------- --------------------
* -05:00 29-MAY-2000 13:14:03
* ALTER SESSION SET TIME\_ZONE = '-8:0';
* SELECT SESSIONTIMEZONE, CURRENT\_DATE FROM DUAL;
* SESSIONTIMEZONE CURRENT\_DATE
* --------------- --------------------
* -08:00 29-MAY-2000 10:14:33
* 
* Current timestamp returns the current date and time in the session time zone.
* CURRENT\_TIMESTAMP returns a timestamp with timezon, where LOCALTIMESTAMP returns a timestamp value
* ALTER SESSION SET TIME\_ZONE = '-5:0';
* ALTER SESSION SET NLS\_DATE\_FORMAT = 'DD-MON-YYYY HH24:MI:SS';
* SELECT SESSIONTIMEZONE, CURRENT\_TIMESTAMP FROM DUAL;
* SESSIONTIMEZONE CURRENT\_TIMESTAMP
* --------------- ---------------------------------------------------
* -05:00 04-APR-00 01.17.56.917550 PM -05:00
* ALTER SESSION SET TIME\_ZONE = '-8:0';
* SELECT SESSIONTIMEZONE, CURRENT\_TIMESTAMP FROM DUAL;
* SESSIONTIMEZONE CURRENT\_TIMESTAMP
* --------------- ----------------------------------------------------
* -08:00 04-APR-00 10.18.21.366065 AM -08:00
* Some ridiculousness with the oracle doc example. They alter the format that current\_timestamp outputs. It actually includes the region by default.  
  + CURRENT\_TIMESTAMP
  + ------------------------------------------------
  + 22-JUN-20 12.58.56.135000000 PM AMERICA/NEW\_YORK
* 
* ALTER SESSION SET TIME\_ZONE = '-5:00';
* SELECT CURRENT\_TIMESTAMP, LOCALTIMESTAMP FROM DUAL;
* CURRENT\_TIMESTAMP LOCALTIMESTAMP
* -------------------------------------------------------------------
* 04-APR-00 01.27.18.999220 PM -05:00 04-APR-00 01.27.19 PM
* ALTER SESSION SET TIME\_ZONE = '-8:00';
* SELECT CURRENT\_TIMESTAMP, LOCALTIMESTAMP FROM DUAL;
* CURRENT\_TIMESTAMP LOCALTIMESTAMP
* ----------------------------------- ------------------------------
* 04-APR-00 10.27.45.132474 AM -08:00 04-APR-00 10.27.451 AM