



# **Laboratorio Oracle**

**Prof. Alessandra Lumini**  
alessandra.lumini@unibo.it

# Credenziali

Password: USER

Login	Nome	Cognome	Mail
USER1	Dario	Bekic	dario.bekic@studio.unibo.it
USER2	Elena	Boschetti	elena.boschetti@studio.unibo.it
USER3	Alessandro	Brasini	alessandro.brasini3@studio.unibo.it
USER4	Alessandro	Cacciaguerra	alessan.cacciaguerr3@studio.unibo.it
USER5	Luca	Carabini	luca.carabini@studio.unibo.it
USER6	Lorenzo	Domeniconi	lorenzo.domeniconi2@studio.unibo.it
USER7	Ettore	Farinelli	ettore.farinelli@studio.unibo.it
USER8	Giacomo	Foschi	giacomo.foschi3@studio.unibo.it
USER9	Raffaello	Fraboni	raffaello.fraboni@studio.unibo.it
USER10	Nicolò	Ghignatti	nicolo.ghignatti@studio.unibo.it
USER11	Enrico	Marchionni	enrico.marchionni@studio.unibo.it
USER12	Raffaele Francesco	Marrazzo	raffaele.marrazzo@studio.unibo.it
USER13	Giacomo	Pierbattista	giacomo.pierbattista@studio.unibo.it
USER14	Filippo	Pracucci	filippo.pracucci@studio.unibo.it
USER15	Alessandro	Ricci	alessandro.ricci47@studio.unibo.it
USER16	Lorenzo	Rigoni	lorenzo.rigoni2@studio.unibo.it
USER17	Giovanni	Rinchiuso	giovanni.rinchiuso@studio.unibo.it
USER18	Hiba	Soufi	hiba.soufi@studio.unibo.it
USER19	Davide	Speziali	davide.speziali@studio.unibo.it
USER20	Pietro	Ventrucci	pietro.ventrucci@studio.unibo.it
USER21	Alessandra	Versari	alessandra.versari2@studio.unibo.it

# Visual studio code

## Oracle Developer Tools for VS Code

host: **si-oracle-11.csr.unibo.it**

Service name: **SISINF**

Port: **1521**



The screenshot displays the Visual Studio Code interface with the Oracle Developer Tools extension. The Oracle Explorer on the left shows the database structure, with a red arrow pointing to the 'Tables' folder. A connection configuration dialog is open, showing the following details:

- Connection Type: Basic
- Database host name: si-oracle-11.csr.unibo.it
- Port number: 1521
- Service name: SISINF
- Connection string: si-oracle-11.csr.unibo.it:1521/SISINF
- Role: Default
- User name: UTENTE0
- Password: .....

The main editor shows a SQL script for creating a table and inserting data:

```
1 create TABLE example_table (  
2   id INT NOT NULL,  
3   name VARCHAR(255) NOT NULL,  
4   PRIMARY KEY (id)  
5 );  
6  
7 INSERT into example_table (id, name) VALUES (1, 'John');  
8 insert into example_table (id, name) VALUES (2, 'Jane');  
9 insert into example_table (id, name) VALUES (3, 'Jack');  
10  
11
```

The bottom panel shows the TERMINALE with the following output:

```
C:\Users\ale\Dropbox\software\node>mongosh %DB_CONNECTION_STRING%;  
"mongosh" non è riconosciuto come comando interno o esterno,  
un programma eseguibile o un file batch.  
  
C:\Users\ale\Dropbox\software\node>  
* Cronologia ripristinata  
  
Microsoft Windows [Versione 10.0.22621.1555]  
(c) Microsoft Corporation. Tutti i diritti riservati.  
  
C:\Users\ale\Dropbox\software\node>  
* Cronologia ripristinata  
  
Microsoft Windows [Versione 10.0.22621.1555]  
(c) Microsoft Corporation. Tutti i diritti riservati.  
  
C:\Users\ale\Dropbox\software\node>
```

# Oracle SQL Developer

- ❑ Download:

- <http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html>

- ❑ Unzip, Run

- ❑ Piattaforme supportate

- Microsoft Windows sia a 32 che a 64 bit, Mac OSX e Linux.

# Oracle SQL Developer - funzionalità

## ❑ Per progettisti

- integrazione di **SQL Developer Data Modeler**, per la progettazione e lo sviluppo di modelli dei dati.
- opzioni di **esportazione e importazione dati**

## ❑ Per DBA

- strumenti di **scheduling** delle operazioni
- strumenti di gestione dei **parametri di configurazione** del database, dei **profili** di sicurezza e auditing e di tutti i **file** (*redo log, control file, archive log, data file*) che compongono il database.
- **SQL Tuning Advisor**, per il **tuning degli statement SQL**
- strumenti a supporto della **migrazione**

## ❑ Per sviluppatori

- **Query Builder** visuale integrato con il corrispondente editor testuale di comandi SQL.

# Caratteristiche di SQL Developer

- ❑ Ambiente per scrivere query SQL, PL/SQL e script
- ❑ Ambiente GUI per creare, cancellare, modificare oggetti Oracle (tabelle, viste, utenti, procedure...)
- ❑ Ambiente di debug per PL/SQL
- ❑ Connessioni multiple
- ❑ Explain Plain

# Ambiente di lavoro

Foglio di Lavoro SQL:

- + per creare un foglio di lavoro associato ad una determinata connessione

Pannello Connessioni:

- + per creare una connessione (segue)

Pannello DBA:

- Visualizza->DBA
- Info per amministratori
- + per creare una connessione

Pannello Output DBMS:

- Visualizza->Output DBMS
- + per abilitare Output comandi:  
DBMS\_OUTPUT.PUT\_LINE()

# Creare una connessione

- ❑ Nome host: **si-oracle-11.csr.unibo.it**
- ❑ SID: **SISINF**
- ❑ Utente: **<nome>** Password: **<pwd>**
- ❑ Porta **1521** (**11521** per l'accesso da esterno)

Nuovo / Seleziona connessione al database

Nome connessione	Dettagli connessione
ord_hr	hr@//si-oracle-11....
SISINF_DBA	SYSTEM@//si-orac...
<b>SISINF_G1</b>	<b>GRUPPO1@//si-or...</b>
TPCD	USERSI@//si-orad...
TPCD_USERSI	USERSI@//si-orad...

Nome connessione: **SISINF\_G1**  
Nome utente: GRUPPO1  
Password: .....

☒ Salva password

**Oracle** Access

Tipo di connessione: Base Ruolo: predefinito

Nome host: si-oracle-11.csr.unibo.it  
Porta: 1521  
☒ SID: SISINF  
☐ Nome servizio

☐ Autenticazione sistema operativo ☐ Autenticazione Kerberos ☐ Connessione proxy

Stato:

Salva Cancella Test Connetti Annulla

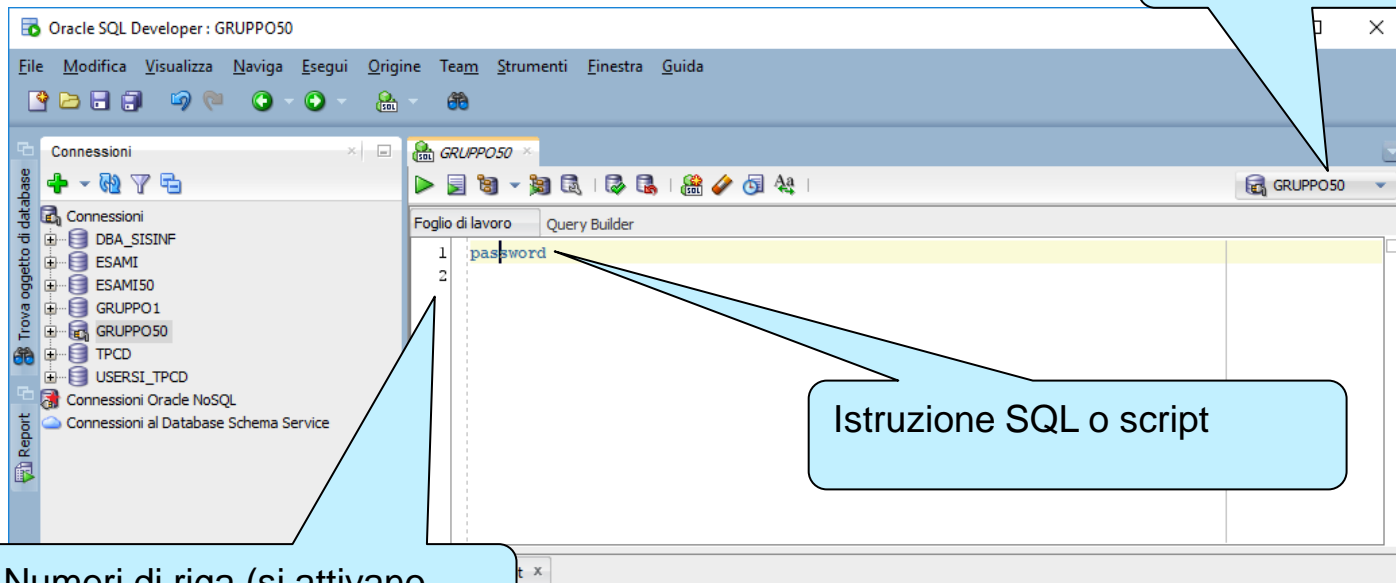


# Esportare una connessione

- ❑ Tasto destro su Connessioni->Esporta
- ❑ Selezionare le connessioni da esportare
- ❑ Scegliere un file **my\_conn.json**
- ❑ Usare chiave di cifratura (o rimuovere password)
- ❑ Per importarla si usa Connessioni->Importa
- ❑ Cambiare la password: digitare **password** nella finestra di comando









# Foglio di lavoro

Connessione attiva



Numeri di riga (si attivano con il tasto destro)

Istruzione SQL o script

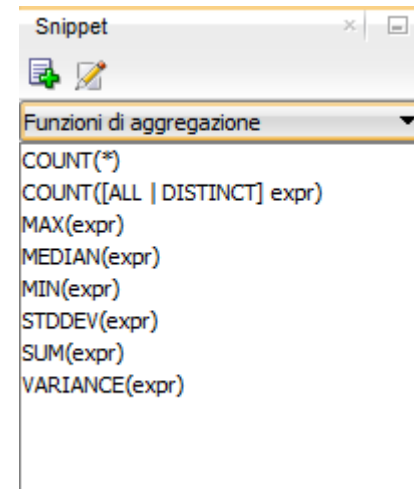
-  – Execute Statement (F9)
-  – Run Script (F5)
-  – Commit (F2)
-  – Rollback (F3)
-  – Cancel (Ctrl+Q)
-  – SQL History (F9)
-  – Execute Explain Plan (F6)
-  – Clear (Ctrl+D)

Esegue l'istruzione evidenziata o su cui è posizionato il cursore. Permette di richiedere variabili bind (precedute da ":").

Esegue più di una istruzione. Il risultato è mostrato nella finestra di script output

# Snippet

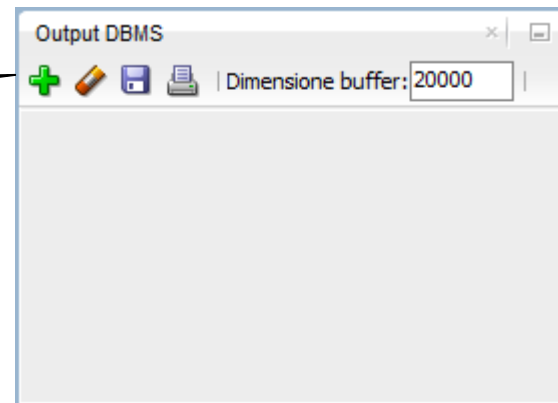
- ❑ SQL developer fornisce molti snippet per aiutare a scrivere codice
- ❑ Menu visualizza->snippets
- ❑ Scegliere la classe e trainare l'istruzione



# DBMS\_OUTPUT.PUT\_LINE

- ❑ L'istruzione DBMS\_OUTPUT.PUT\_LINE può essere usata per scrivere in output valori di controllo o messaggi di errore utili in fase di debug
- ❑ L'output viene mostrato nella finestra DBMS\_OUTPUT (Visualizza->Output DBMS)
- ❑ DBMS\_OUTPUT deve essere abilitato per la connessione corrente prima di lanciare i test (tramite pulsante o con istruzione DBMS\_OUTPUT.ENABLE)

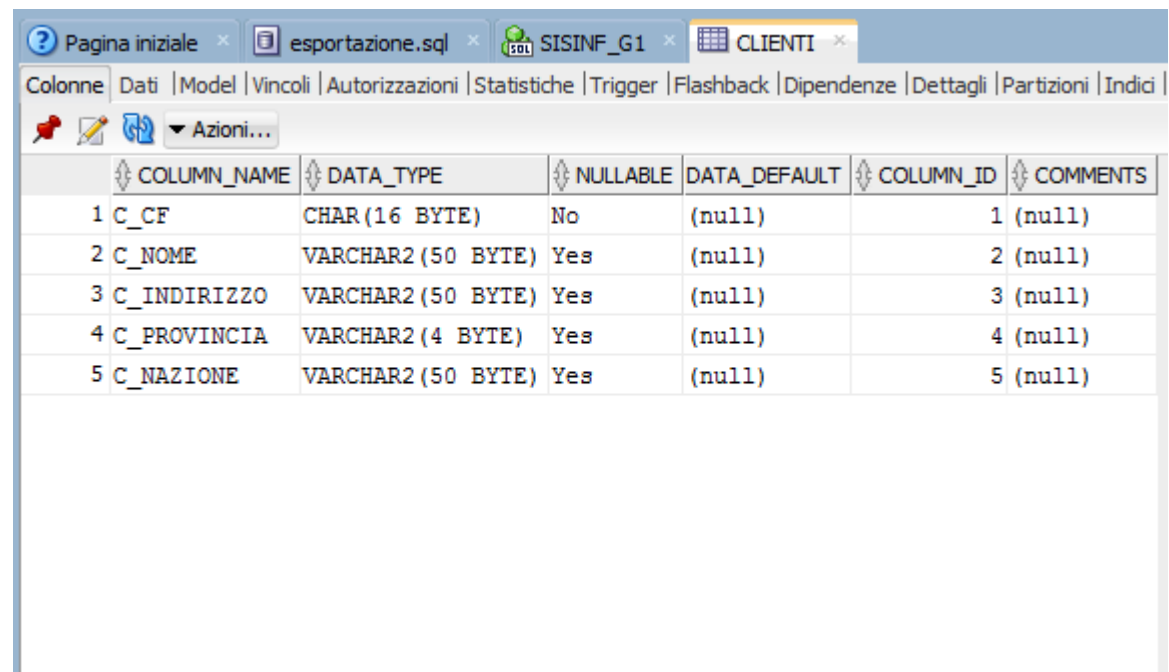
Abilitare DBMS\_OUTPUT



# Visualizzare una Tabella

❑ Selezionando una tabella si apre un pannello multi-tab che permette di visualizzarne:

- Struttura
- Dati
- Modello
- Vicoli
- Statistiche
- Trigger
- ....
- Istruzioni SQL



The screenshot shows a database management interface with a multi-tabbed window. The active tab is labeled 'CLIENTI'. Below the tabs, there is a menu bar with options: 'Colonne', 'Dati', 'Model', 'Vincoli', 'Autorizzazioni', 'Statistiche', 'Trigger', 'Flashback', 'Dipendenze', 'Dettagli', 'Partizioni', and 'Indici'. A toolbar with icons for various actions is visible. The main area displays a table structure with the following columns: COLUMN\_NAME, DATA\_TYPE, NULLABLE, DATA\_DEFAULT, COLUMN\_ID, and COMMENTS. The table contains five rows of data.

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	C_CF	CHAR(16 BYTE)	No	(null)	1	(null)
2	C_NOME	VARCHAR2(50 BYTE)	Yes	(null)	2	(null)
3	C_INDIRIZZO	VARCHAR2(50 BYTE)	Yes	(null)	3	(null)
4	C_PROVINCIA	VARCHAR2(4 BYTE)	Yes	(null)	4	(null)
5	C_NAZIONE	VARCHAR2(50 BYTE)	Yes	(null)	5	(null)

# Modificare una tabella

- ❑ Con il menù contestuale Modifica si apre un pannello per editarne lo schema

Modifica Tabella

Schema: GRUPPO1

Nome: CLIENTI

Tipo di tabella: Normale

Ricerca

Colonne

- Vincoli
- Indici
- Memorizzazione
- Commento
- DDL

Colonne: nome

PK	Nome	Tipo di dati	Dimensione	Non nullo	Predefinito	Commento
	C_CF	CHAR	16	<input checked="" type="checkbox"/>		
	C_NOME	VARCHAR2	50	<input type="checkbox"/>		
	C_INDIRIZZO	VARCHAR2	50	<input type="checkbox"/>		
	C_PROVINCIA	VARCHAR2	4	<input type="checkbox"/>		
	C_NAZIONE	VARCHAR2	50	<input type="checkbox"/>		

Tipo dati Vincoli Indici Parametri LOB Colonna identità

OK Annulla

# Esportare una tabella

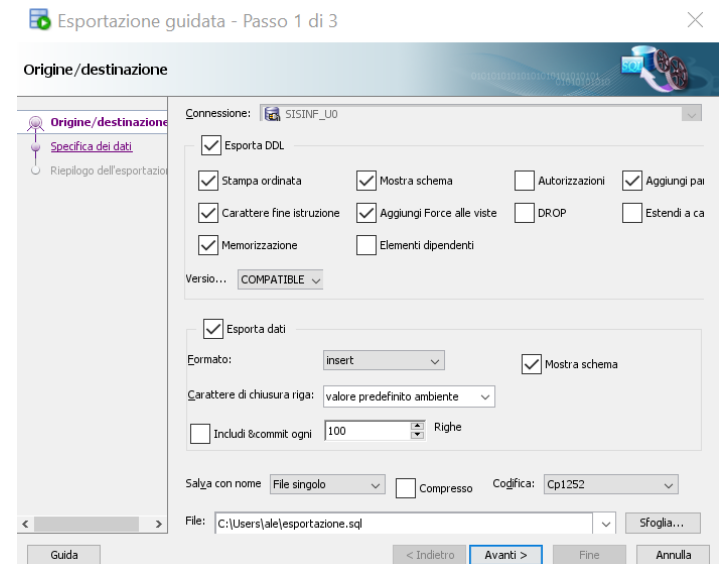
- ❑ Con il menù contestuale Esporta si apre un pannello per esportare sia lo schema che i dati in formato SQL
  - Esportare schema (DDL) e dati (DML)
  - Selezionare file di output

## ❑ Import

- CSV, TXT, XLS da menu
- SQL apri file e esegui

## ❑ DB Dump

- Operazione complessa che richiede permessi da amministratore
- Vedere Oracle Data Pump sui manuali



# Esercizi

- A. Creare le seguenti tabelle:

LA\_VENDITE(V\_Num,Prodotti:V\_Prodotto,V\_Data,V\_Qta)

LA\_PRODOTTI(P\_Cod,P\_Nome,P\_QtaDisp,P\_PrezzoList)

- B. Aggiungere la tupla alla tabella LA\_PRODOTTI

P01,Vite,1000, 0.05

- C. Creare un indice sull'attributo P\_Nome della tabella LA\_PRODOTTI memorizzandolo nel tablespace INDX.



# Tips

- Creazione di tabelle ( vedi 16.6 SQL Language Reference)  
CREATE TABLE *nometabella* (  
    *nomecol* DATATYPE,  
    *nomecol* DATATYPE,  
    PRIMARY KEY (*nomecol*,...,*nomecol*)  
    [,FOREIGN KEY (*nomecol*,...,*nomecol*)  
    REFERENCES *nometabella* (*nomecol*,...,*nomecol*)];  
    dove DATATYPE è uno dei tipi di dati previsti da ORACLE
- Inserimento valori ( vedi 18.54 SQL Language Reference)  
INSERT INTO *nometabella*  
    VALUES (*valore*,...,*valore*);
- Creazione di indici (vedi 14.61 SQL Language Reference)  
CREATE INDEX [UNIQUE,BITMAP] *nomeindice*  
    ON *nometabella* (*nomecol*,...,*nomecol*);

# Utenti e schemi

- ❑ Lo **schema** è un insieme di tabelle ed oggetti di altro tipo di proprietà di un utente.
- ❑ L'elenco degli utenti si può vedere nella sezione **Altri utenti** di una connessione:
  - L'utente SYS è il superutente, proprietario di tutti gli oggetti di sistema, usato solo per operazioni di amministrazione.
  - L'utente SYSTEM è un utente speciale, ha ampia delega da parte di SYS ad eseguire la maggior parte delle attività di amministrazione.
  - ..
- ❑ Se abbiamo gli opportuni permessi possiamo **interrogare le tabelle** di altri utenti.
- D. Mostrare utente corrente
- E. Interrogare una tabella dell'utente **nw**
  - Sono riferibili con la notazione **nw.<nometabella>**

# Il dialetto PL/SQL

- ❑ SQL su Oracle differisce in alcuni costrutti dallo standard
  - ❑ Gestione delle date
  - ❑ Table expression
  - ❑ Query Top K
  - ❑ Statement Case
  - ❑ ...
- 
- ❑ Si vedano gli esercizi su 0\_LabSQL per un ripasso del linguaggio SQL base

# Schema DB



# L'istruzione SELECT

- ❑ È l'istruzione che permette di eseguire interrogazioni (*query*) sul DB

```
SELECT  A1, A2, ..., Am
FROM    R1, R2, ..., Rn
[WHERE  <condizione>]
[GROUP BY <listaAttributi>]
[HAVING <condizione>]
[ORDER BY <listaAttributi>]
```

➤ ovvero:

- SELECT (o TARGET) list (cosa si vuole come risultato)
- clausola FROM (da dove si prende)
- clausola WHERE (che condizioni deve soddisfare)
- clausola GROUP BY (le colonne su cui raggruppare)
- clausola HAVING (condizione relative ai gruppi)
- clausola ORDER BY (ordinamento)

Il comando **SELECT** permette di realizzare le operazioni di selezione, proiezione, join, raggruppamento e ordinamento.

# Esempio di query complessa

- ❑ Selezionare gli impiegati che hanno gestito più di 50 ordini

```
SELECT A.LASTNAME, A.FIRSTNAME, count(*) as NumeroOrdini  
FROM NW.ORDERS B, NW.EMPLOYEES A  
WHERE A.EMPLOYEEID = B.EMPLOYEEID  
group by A.EMPLOYEEID, A.LASTNAME, A.FIRSTNAME  
having count(*) > 50  
order by 3 desc;
```

1. Selezionare i prodotti che sono stati venduti a più di 10 clienti diversi

# Gestione delle date

## ❑ Funzioni di conversione Stringa-Data

**TO\_DATE( stringa [, formato] )**

```
TO_DATE('2003/07/09', 'yyyy/mm/dd')  
TO_DATE('070903', 'MMDDYY')  
TO_DATE('20020315', 'yyyymmdd')
```

*Result: date value of July 9, 2003*  
*Result: date value of July 9, 2003*  
*Result: date value of Mar 15, 2002*

**TO\_CHAR( value [, format\_mask] )**

```
TO_CHAR(sysdate, 'yyyy/mm/dd')  
TO_CHAR(sysdate, 'Month DD, YYYY')  
TO_CHAR(sysdate, 'YYYY')
```

*Result: '2003/07/09'*  
*Result: 'July 09, 2003'*  
*Result: '2003'*

## ❑ Selezionare gli ordini dal 1/1/1998 al 1/2/1998

```
SELECT ORDERID, ORDERDATE  
FROM NW.ORDERS  
WHERE ORDERDATE  
BETWEEN TO_DATE ('01/01/1998', 'dd/mm/yyyy')  
AND TO_DATE ('1998/02/01', 'yyyy/mm/dd');
```

- NB. **TO\_DATE()** non serve se il formato è quello specificato nel parametro **NLS\_DATE\_FORMAT**

## 2. Selezionare gli impiegati che hanno più di 60 anni

# Gestione delle date

## ❑ Funzione di EXTRACT

```
EXTRACT (  
{ YEAR | MONTH | DAY }  
FROM date_value )
```

EXTRACT(YEAR FROM DATE '2003-08-22')	Result: 2003
EXTRACT(MONTH FROM DATE '2003-08-22')	Result: 8
EXTRACT(DAY FROM DATE '2003-08-22')	Result: 22

## ❑ Ordinare gli impiegati per anno di nascita (dal più giovane)

```
SELECT EMPLOYEEID, LASTNAME, FIRSTNAME, EXTRACT( YEAR FROM BIRTHDATE)  
FROM NW.EMPLOYEES  
ORDER BY 4 DESC;
```

```
SELECT EMPLOYEEID, LASTNAME, FIRSTNAME, to_char(BIRTHDATE, 'YYYY')  
FROM NW.EMPLOYEES  
ORDER BY 4 DESC;
```

3. Selezionare gli ordini per cui l'anno di spedizione è diverso da quello di ordine



# Table expressions

- ❑ Table expressions = viste create dinamicamente in memoria su cui si possono poi effettuare delle query.

```
WITH <alias_name> AS (subquery_sql_statement)
SELECT <column_name_list> FROM <alias>;
```

- ❑ Selezionare l'impiegato che ha gestito il maggior numero di ordini

```
WITH imp1 AS (SELECT LASTNAME, FIRSTNAME, count(ORDERID) as NumOrdini
FROM NW.EMPLOYEES E join NW.ORDERS o on (e.EMPLOYEEID=o.EMPLOYEEID)
group by e.EMPLOYEEID, e.LASTNAME, e.FIRSTNAME
order by 3 desc)
```

```
SELECT LASTNAME, FIRSTNAME, NumOrdini
FROM imp1
where NumOrdini>=ALL(SELECT NumOrdini FROM imp1);
```

- ❑ Se si vogliono usare più table expression non bisogna ripetere WITH ma separarle con virgole

# Table expressions

Oppure si può formulare la query direttamente nella clausola FROM (ma è meno leggibile)

```
SELECT <column_name_list> FROM (<query>);
```

```
SELECT LASTNAME, FIRSTNAME, NumOrdini
FROM (SELECT LASTNAME, FIRSTNAME, count(ORDERID) as NumOrdini
FROM NW.EMPLOYEES E join NW.ORDERS o on (e.EMPLOYEEID=o.EMPLOYEEID)
group by e.EMPLOYEEID, e.LASTNAME, e.FIRSTNAME
order by 3 desc)
where NumOrdini>=ALL(SELECT NumOrdini FROM (SELECT LASTNAME, FIRSTNAME,
count(ORDERID) as NumOrdini
FROM NW.EMPLOYEES E join NW.ORDERS o on (e.EMPLOYEEID=o.EMPLOYEEID)
group by e.EMPLOYEEID, e.LASTNAME, e.FIRSTNAME
order by 3 desc));
```

4. Selezionare il cliente che ha effettuato il maggior numero di ordini
5. Selezionare il cliente che ha generato il più alto introito nel 1996

# ROWNUM

- ❑ La funzione ROWNUM restituisce l'ordinale di una riga in una tabella (la prima riga ha ROWNUM 1, ecc.)
- ❑ Se la tabella è ordinata si può filtrare su ROWNUM per selezionare il primo valore
- ❑ Selezionare l'impiegato che ha gestito il maggior numero di ordini

```
SELECT EMPLOYEEID, LASTNAME, FIRSTNAME, NumOrdini
FROM (SELECT e.EMPLOYEEID, e.LASTNAME, e.FIRSTNAME, count(ORDERID) as NumOrdini
FROM NW.EMPLOYEES E join NW.ORDERS o on (e.EMPLOYEEID=o.EMPLOYEEID)
group by e.EMPLOYEEID, e.LASTNAME, e.FIRSTNAME
order by 4 desc)
where rownum<=1;
```

```
WITH imp1 AS (SELECT LASTNAME, FIRSTNAME, count(ORDERID) as NumOrdini
FROM NW.EMPLOYEES E join NW.ORDERS o on (e.EMPLOYEEID=o.EMPLOYEEID)
group by e.EMPLOYEEID, e.LASTNAME, e.FIRSTNAME
order by 4 desc)
SELECT LASTNAME, FIRSTNAME, NumOrdini
FROM imp1
where rownum<=1;
```

# Select Top K in Oracle

- ❑ **TOP K** (MSSql) e **Limit** (MySQL) non esistono in Oracle

```
select a, b from tabella where rownum<K order by b;  
--non funziona perchè rownum è valutato prima di ordinare
```

- ❑ 2 Soluzioni (con Table expression):

```
select a, b  
from (select a, b from tabella order by b)  
where rownum<K;
```

```
select a, b  
from (select a, b, rank() over (order by b) r from tabella)  
where r<K;
```

- ❑ La funzione **RANK** restituisce l'ordine in un gruppo di valori (vedi anche **DENSE\_RANK**)

# Select Top K in Oracle 12

- ❑ Ulteriore soluzione (con **Fetch first K rows only**):

```
select a, b  
from (select a, b from tabella order by b)  
Fetch first K rows only;
```

- ❑ Sintassi:

[ OFFSET offset ROWS]

FETCH NEXT [ row\_count | percent PERCENT ] ROWS

[ ONLY | WITH TIES ]

- La clausola **OFFSET** specifica il numero di righe da saltare prima che inizi la limitazione delle righe.
- **ONLY** restituisce esattamente il numero di righe richieste, **WITH TIES** restituisce ulteriori righe con la stessa chiave di ordinamento dell'ultima riga recuperata.

# Select Top K

- ❑ Selezionare i 10 ordini con valore più alto

```
select ORDERID, Totale from  
  (select ORDERID, sum (UNITPRICE*QUANTITY*(1-DISCOUNT)) as Totale  
   from NW.ORDERDETAILS  
   group by ORDERID  
   order by 2 desc)  
where rownum<=10;
```

6 Selezionare i 5 clienti che hanno effettuato il maggior numero di ordini

# CASE Statement

- Nella SELECT LIST:

```
CASE [ expression ]  
  WHEN condition_1 THEN result_1  
  ...  
  WHEN condition_n THEN result_n  
  ELSE result  
END
```

- Raggruppare i prodotti in fasce in base alle quantità venduta: fascia alta (>1000), fascia bassa (<500), fascia media

```
with PView as (SELECT PRODUCTID, sum (o.quantity) as QtaVenduta  
FROM NW.ORDERDETAILS O  
group by PRODUCTID)  
select PRODUCTID, case  
  when QtaVenduta>1000 then 'fascia alta'  
  when QtaVenduta<500 then 'fascia bassa'  
  else 'fascia media'  
end fascia, QtaVenduta  
from PView;
```

7 Modificare le soglie in  $(s*2/3)$ ,  $(s*1/3)$ , dove  $s$  è la quantità venduta dal prodotto più venduto.

# Costrutti avanzati: Window Functions

- ❑ Le funzioni di aggregazione (**SUM()**, **COUNT()**..) o il **GROUP BY** riducono il numero di righe restituite dalla query.
- ❑ Come le funzioni aggregate con la clausola GROUP BY, anche le **Window Functions** operano su un sottoinsieme di righe, ma NON riducono il numero di righe restituite dalla query.

```
SELECT productname, categoryid, unitprice,  
MAX(unitprice) OVER (PARTITION BY categoryid) AS MaxCatPrice,  
MAX(unitprice) OVER () AS MaxPrice  
FROM nw.products;
```

In questo esempio, la funzione **MAX()** funziona come una **funzione finestra** che opera su un insieme di righe definito dal contenuto della clausola OVER.

PName	CatID	UnitPrice	MaxCatPrice	MaxPrice
Aniseed Syrup	2	10	43,9	263,5
Boston Crab Meat	8	18,4	62,5	263,5
Camembert Pierrot	4	34	55	263,5
Carnarvon Tigers	8	62,5	62,5	263,5
Chai	1	18	263,5	263,5
Chang	1	19	263,5	263,5
Chartreuse verte	1	18	263,5	263,5
Chef Anton's Cajun	2	22	43,9	263,5



# Window Functions: sintassi

```
window_function_name(expression) OVER (  
  [PARTITION BY expression]  
  [ORDER BY expression]  
  [windowing_clause]  
)
```

- ❑ La clausola **PARTITION BY** suddivide le righe in partizioni.
- ❑ La clausola **ORDER BY** specifica come le righe sono ordinate all'interno di una partizione.
- ❑ Una **windowing\_clause** permette di selezionare un sottoinsieme della partizione corrente definito rispetto alla riga corrente.
- ❑ Si noti che le funzioni finestra vengono eseguite sul set di risultati **dopo** tutte le clausole JOIN, WHERE, GROUP BY e HAVING e **prima** di ORDER BY, LIMIT e SELECT DISTINCT.
- ❑ Alcune Window Functions in Oracle
  - Funzioni di aggregazione: SUM, AVG, MIN, MAX, COUNT, VARIANCE, STDDEV
  - Funzioni di ranking: ROW\_NUMBER, RANK, DENSE\_RANK, PERCENT\_RANK, CUME\_DIST
  - Funzioni di offset e di spostamento: LEAD, LAG, FIRST\_VALUE, LAST\_VALUE, NTH\_VALUE
  - Funzioni di stringa: NTILE, GROUPING

# Window Functions: esempio

- ❑ Per ogni categoria selezionare i cinque prodotti più costosi

```
WITH ranked_prod AS  
(SELECT categoryid, productname, unitprice,  
  RANK() OVER (PARTITION BY categoryid ORDER BY unitprice DESC) rnk  
FROM nw.products  
ORDER BY 1,4)
```

```
SELECT * FROM ranked_prod  
WHERE rnk < 6;
```

8. Per ogni categoria selezionare il costo medio relativo ai cinque prodotti più costosi