Database Management

COMP 3010E FALL 2025

LECTURE 6 INTRODUCTION TO SQL

Introduction to SQL

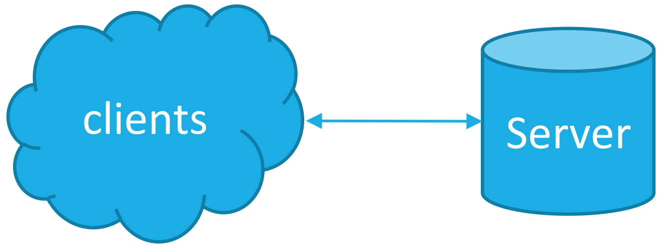
- **Structured Query Language** (SQL)
  - Most common and *standard* language for creating and querying relational databases
  - Set-based declarative programming language
  - Run on machines of all sizes from personal computers to large mainframes
  - Standard by ANSI, FIPS, and ISO
  - Many variations from vendor to vendor

History of SQL

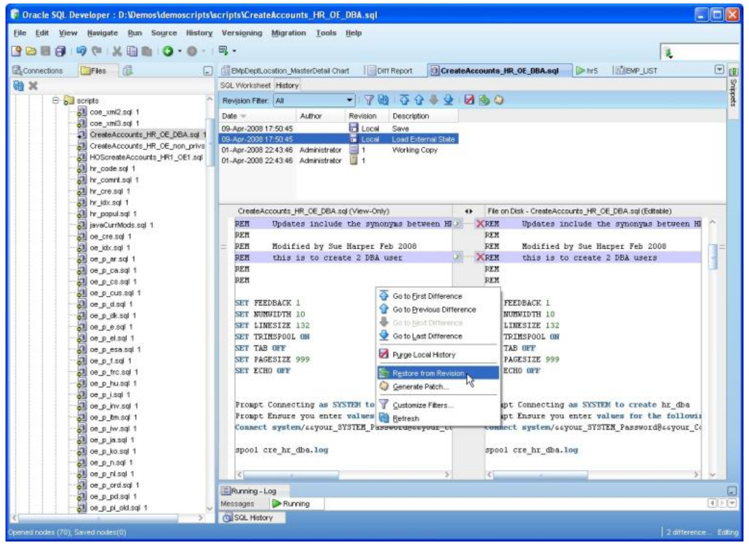
- First implemented on System R (IBM) named Sequel, 1975-1979
- SQL-86 “Structured Query Language”
- SQL-89: ANSI / ISO SQL standard (SQL 1)
- SQL-92 (SQL 2): More data types, operations, integrity constraints, etc.
- SQL-99 (SQL 3): Core SQL + SQL extension: Object-oriented, abstract data types, triggers
- Additional standards.

Types of DBMS Software

- Two Tier Database Systems
  - Multi-user access



A program you can launch from any computer



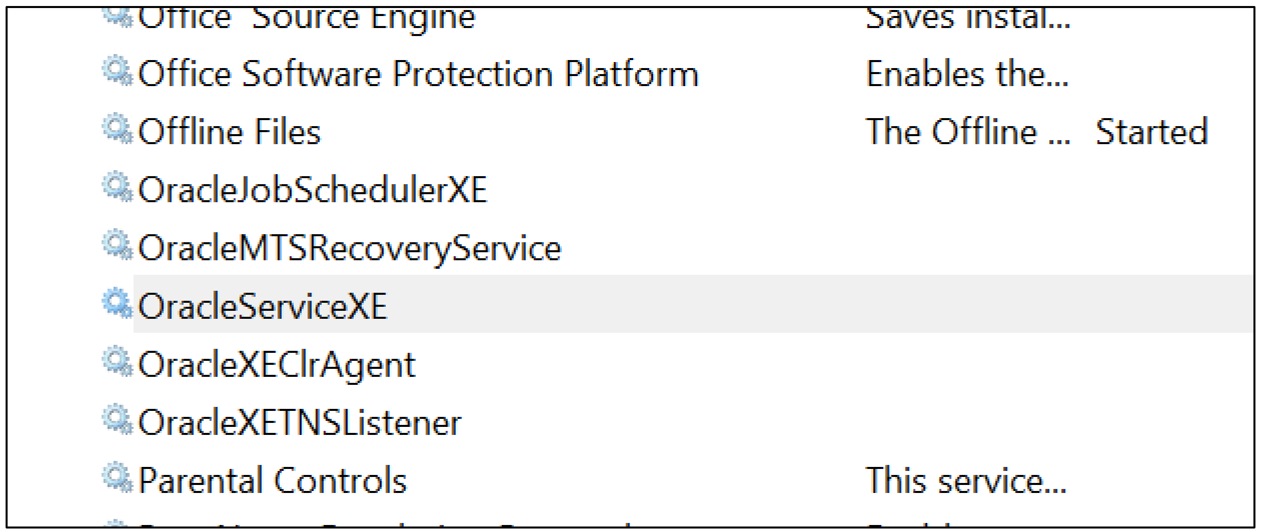
```
REM     Updates include the synonyms between HR
REM
REM     Modified by Sue Harper Feb 2008
REM     this is to create 2 DBA users
REM
REM

SET FEEDBACK 1
SET NUMWIDTH 10
SET LINESIZE 132
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 999
SET ECHO OFF

Prompt Connecting as SYSTEM to create hr_dba
Prompt Ensure you enter values for the following
Connect system/&&your_SYSTEM_Password@&&your_Co

spool cre_hr_dba.log
```

Run as a service on a server

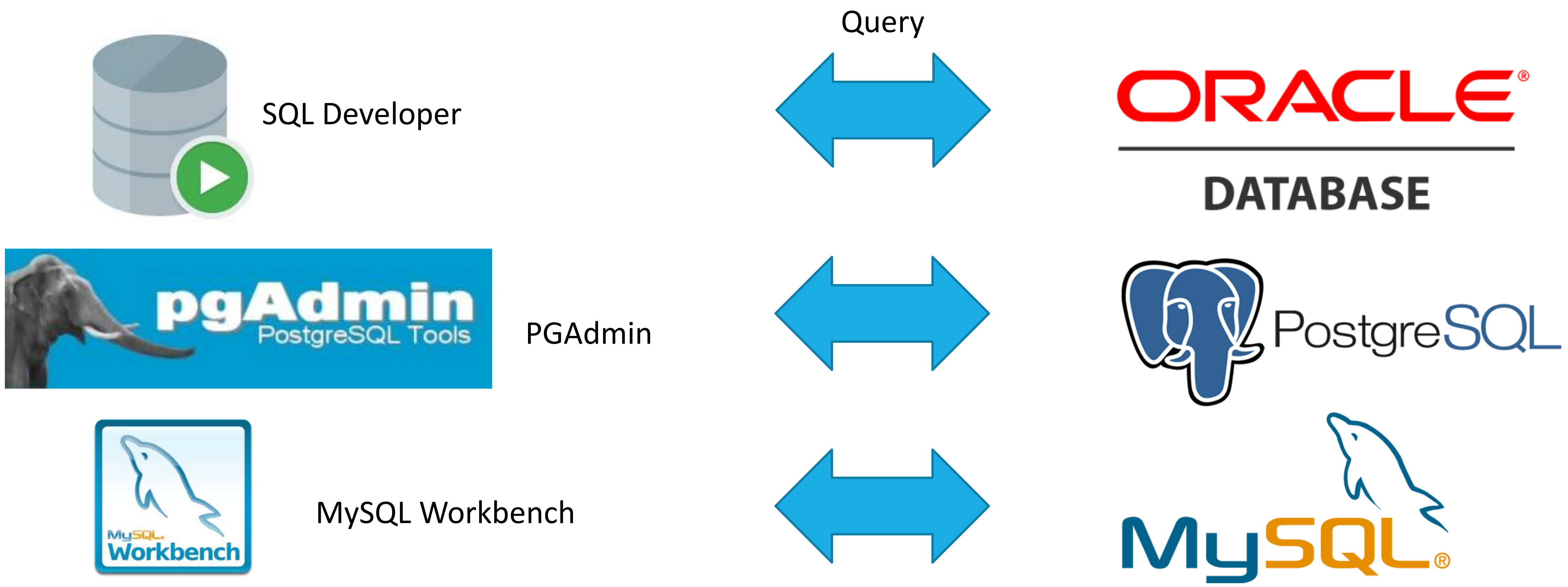


Office Source Engine	Saves instal...	
Office Software Protection Platform	Enables the...	
Offline Files	The Offline ...	Started
OracleJobSchedulerXE		
OracleMTSRecoveryService		
OracleServiceXE		
OracleXEClrAgent		
OracleXETNSListener		
Parental Controls	This service...	

Two-Tier Database

Client

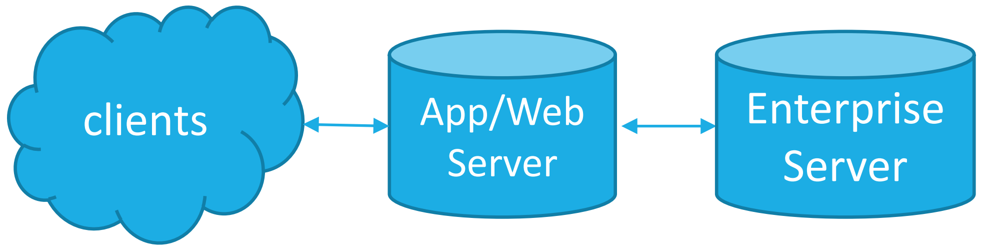
DBMS Software



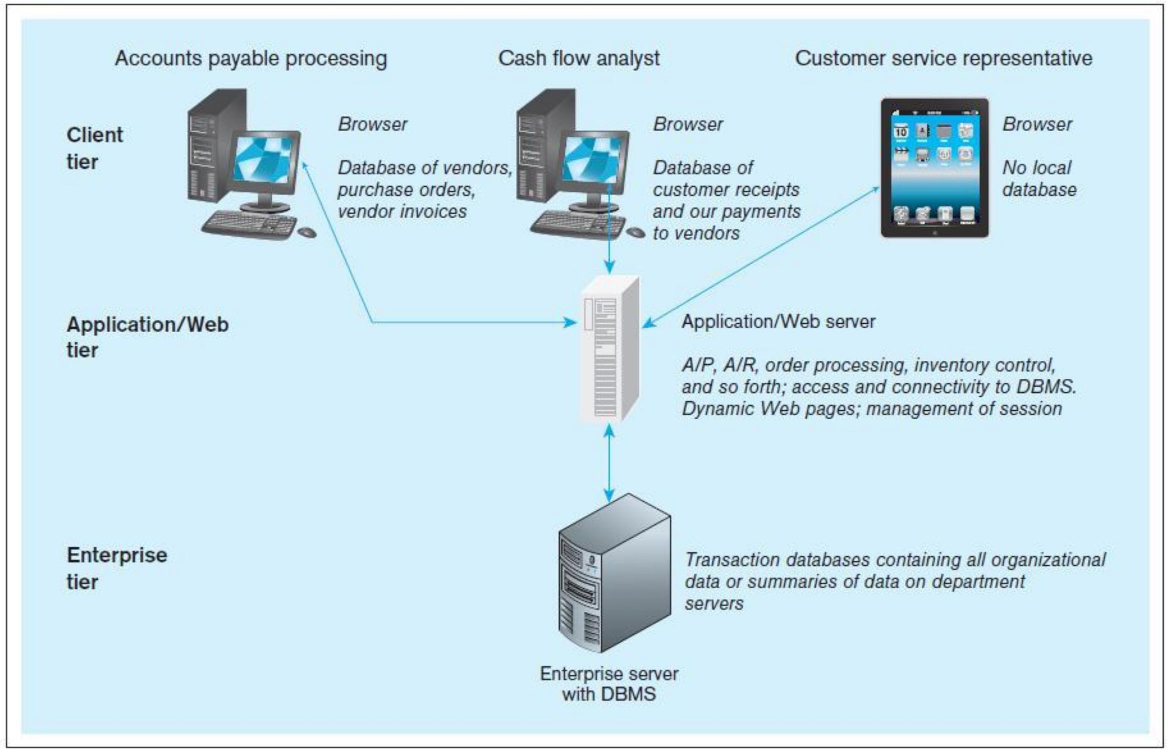
Range of Database Applications

Multi-tier Client/Server Database

- Large number of users in a department/division
- Client runs user interface only
- Application/Web server contains business logic
- Enterprise server runs DBMS and databases
- Example: web app., cloud computing…

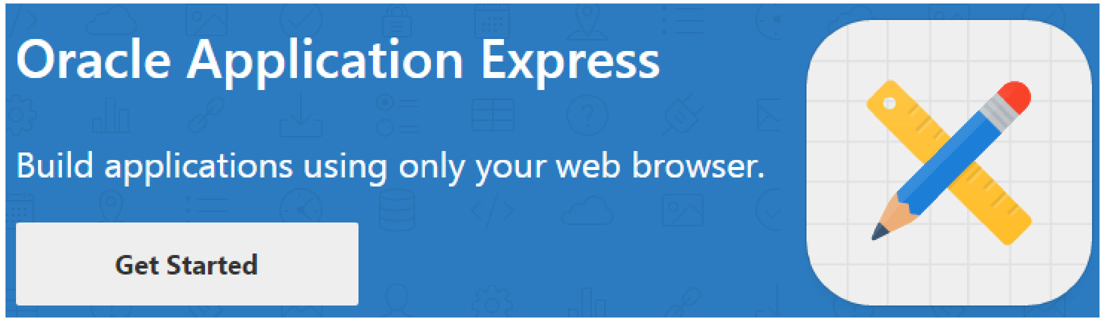


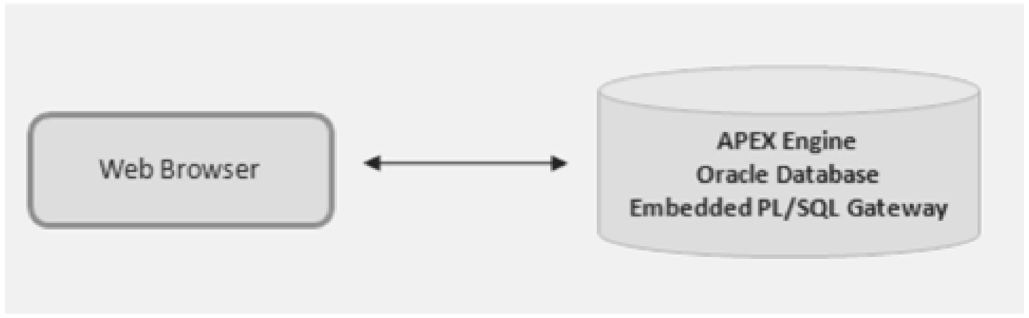
Cloud-based DBMS!


Accounts payable processing
Cash flow analyst
Customer service representative
Client tier
Browser
Database of vendors, purchase orders, vendor invoices
Browser
Database of customer receipts and our payments to vendors
Browser
No local database
Application/Web tier
Application/Web server
A/P, A/R, order processing, inventory control, and so forth; access and connectivity to DBMS. Dynamic Web pages; management of session
Enterprise tier
Transaction databases containing all organizational data or summaries of data on department servers
Enterprise server with DBMS

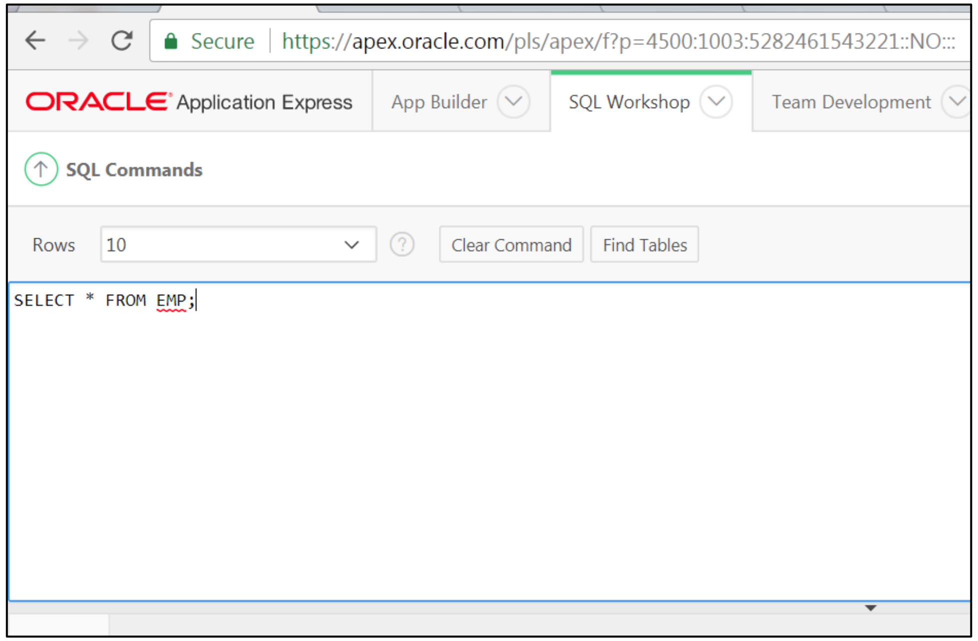
APEX

- **Oracle Application Express (APEX)**
  - A web-based application development tool
  - Integrated with the Oracle database
  - Enables you to design, develop and deploy responsive, database-driven applications using only your web browser
  - We use APEX for course project (https://apex.oracle.com/en/)


Oracle Application Express
Build applications using only your web browser.
Get Started

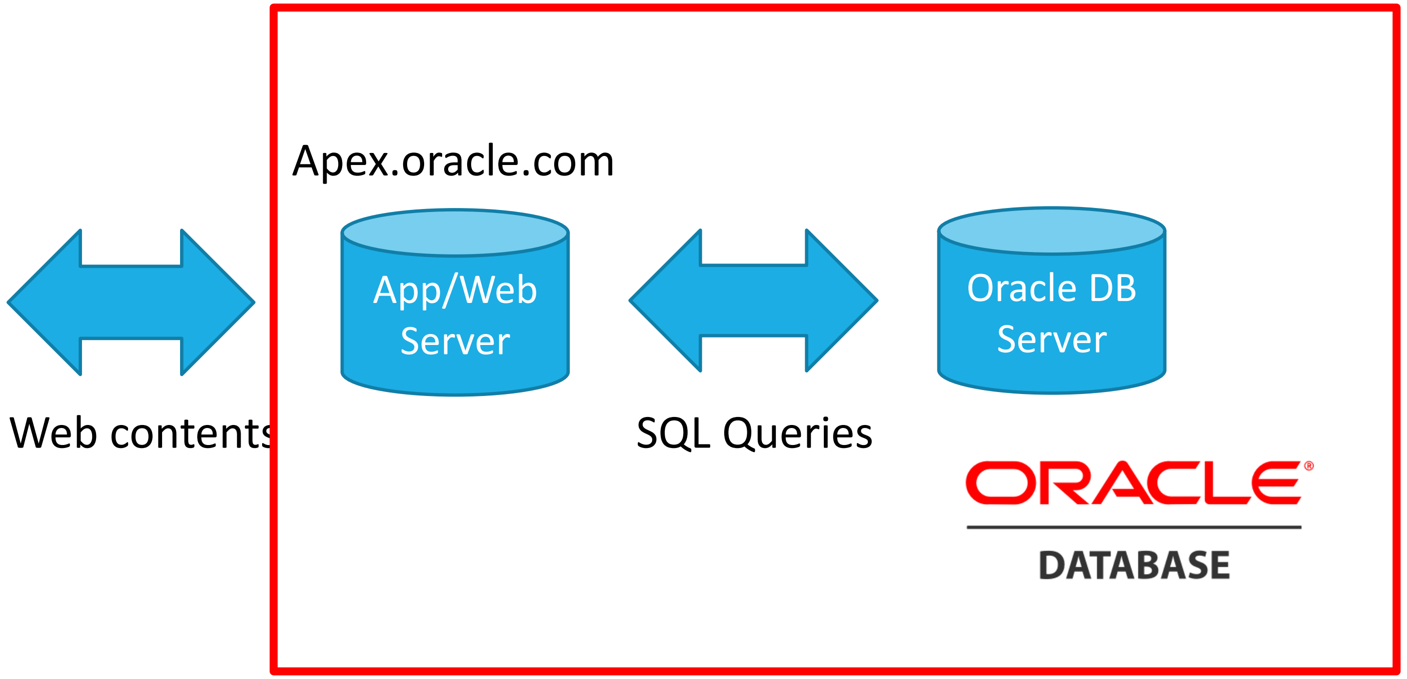


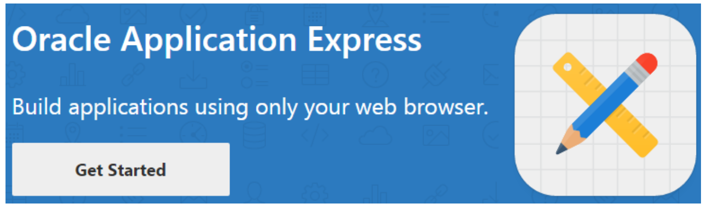
3-Tier Database: APEX


Secure
https://apex.oracle.com/pls/apex/f?p=4500:1003:5282461543221::NO:::
ORACLE Application Express
App Builder
SQL Workshop
Team Development
SQL Commands
Rows
10
Clear Command
Find Tables

```
SELECT * FROM EMP;
```

User's Browser

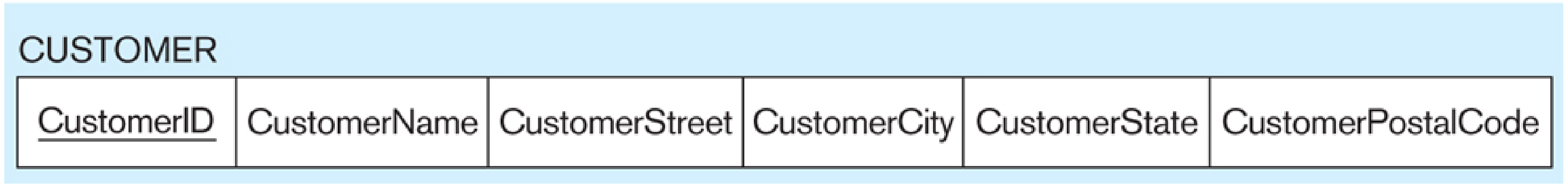


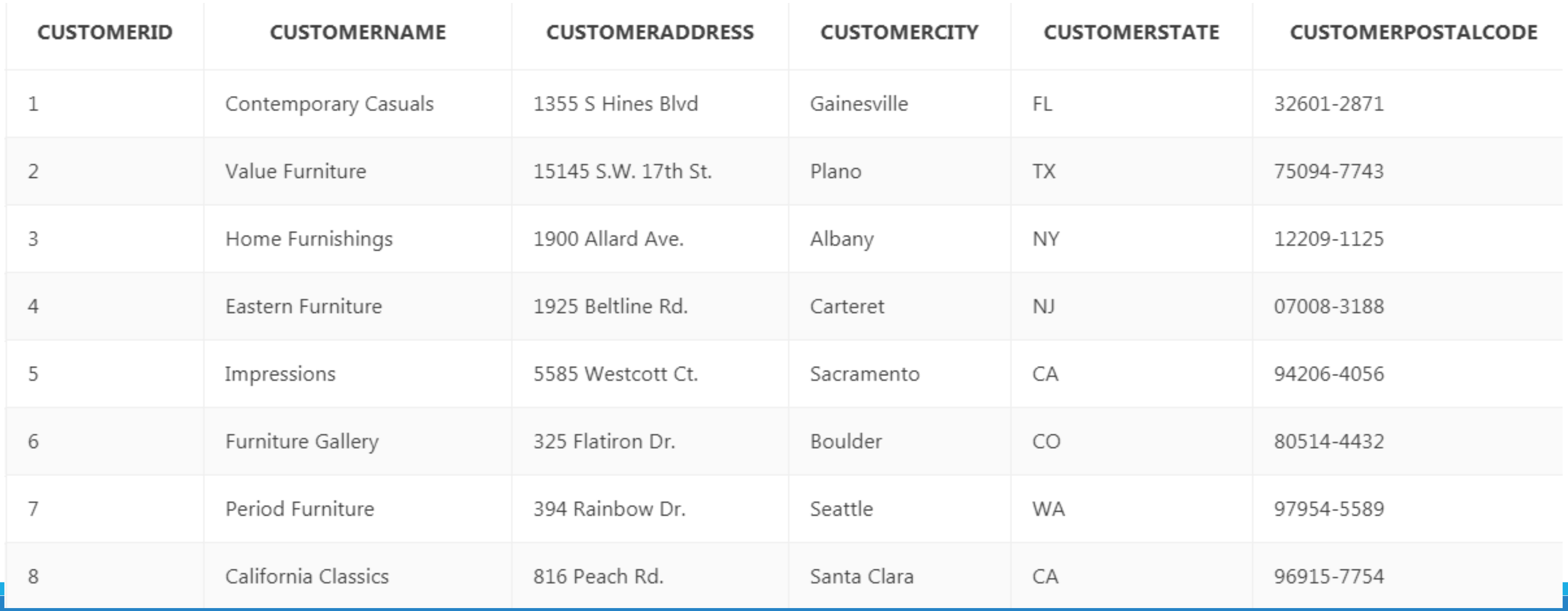

Oracle Application Express
Build applications using only your web browser.
Get Started

Types of SQL Commands

- **Data Definition Language (DDL)** commands
  - Create, alter, and drop tables, views, and indexes
- **Data Manipulation Language (DML)** commands
  - Core commands of SQL
  - Insert, update, and query the data
  - Issued interactively or embedded in a PL (e.g. Java, PHP)
- **Data Control Language (DCL)** commands
  - Set up access and controls for DB administration
  - Grant/revoke privileges and commit data

DDL: Data Definition Language





CUSTOMERID	CUSTOMERNAME	CUSTOMERADDRESS	CUSTOMERCITY	CUSTOMERSTATE	CUSTOMERPOSTALCODE
1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-2871
2	Value Furniture	15145 S.W. 17th St.	Plano	TX	75094-7743
3	Home Furnishings	1900 Allard Ave.	Albany	NY	12209-1125
4	Eastern Furniture	1925 Beltline Rd.	Carteret	NJ	07008-3188
5	Impressions	5585 Westcott Ct.	Sacramento	CA	94206-4056
6	Furniture Gallery	325 Flatiron Dr.	Boulder	CO	80514-4432
7	Period Furniture	394 Rainbow Dr.	Seattle	WA	97954-5589
8	California Classics	816 Peach Rd.	Santa Clara	CA	96915-7754

Common Data Types

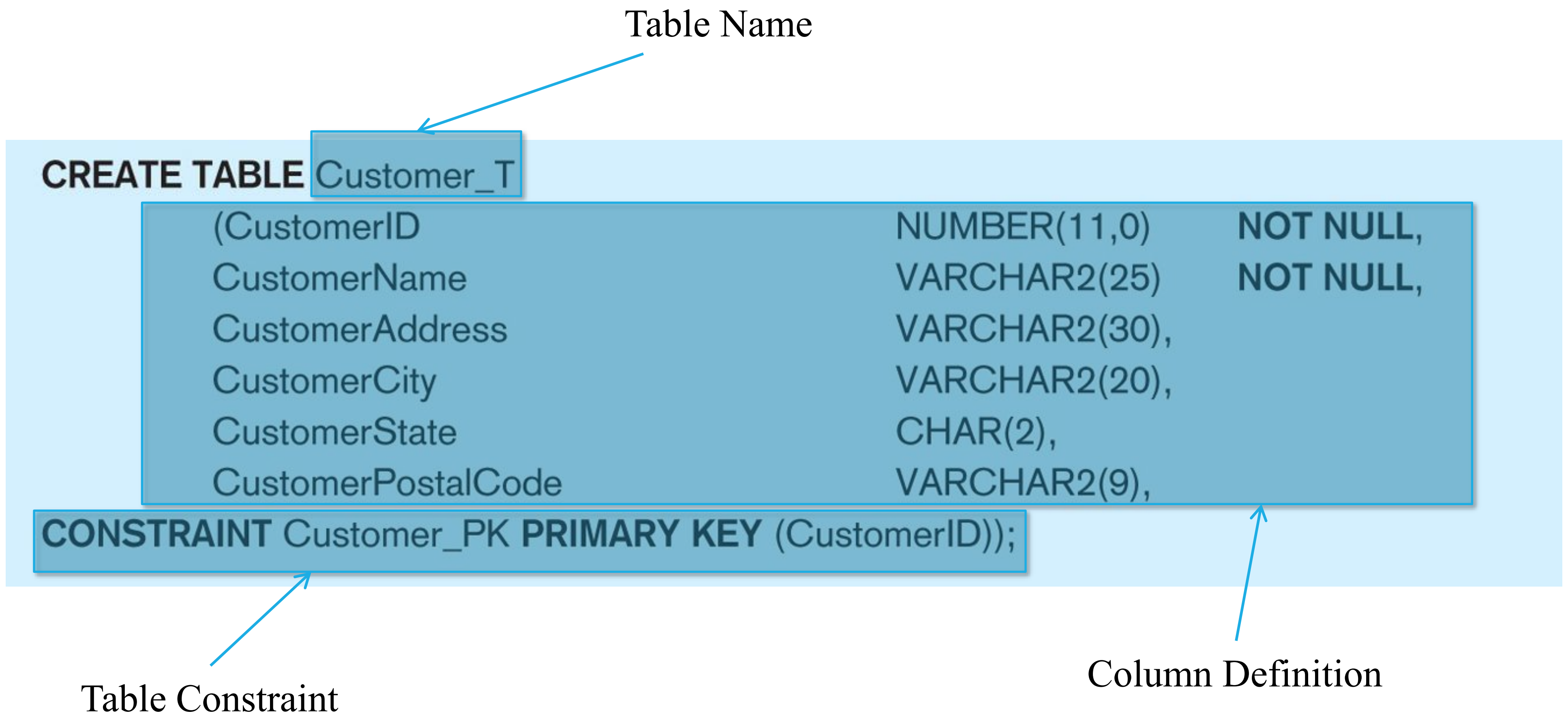
Type	Description	Example
CHAR(x)	Fixed length string with x characters	char(2): 'NY', 'IA', 'CA', 'FL' …
VARCHAR2(x)	Text string with varying length up to x characters	varchar2(20): 'abc', 'John Smith' …
NUMBER	A fixed and floating-point number up to 38 digits of precision	1,2,3, 105.2, 16384.995 …
NUMBER(p, s)	A number with precision p and scale s (p/s: number of digits before/after the decimal)	NUMBER(3, 1): 123.4, 456.9… NUMBER(5, 0): 1, 12, 123, 12345 …
DATE	Date and Time information formatted. Default format: YY-MM-DD	Today is : '16-09-13'
TIMESTAMP	"YYYY-MM-DD HH24:MI:SS"	'2016-01-27 15:30:00'
BOOLEAN	Truth values	TRUE, FALSE, or UNKNOWN

What data type do you use for the phone number or zip code?

Keywords in SQL

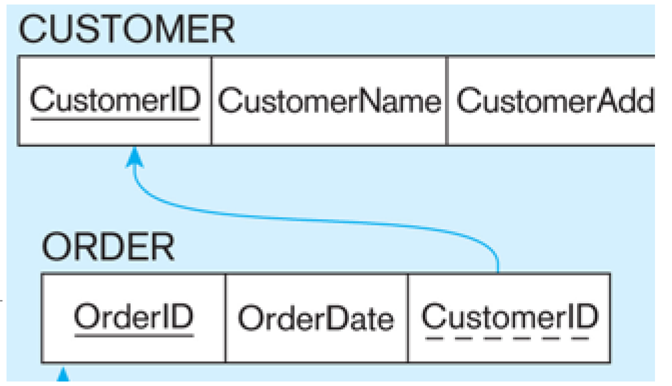
- Table names, column names, keywords are NOT case sensitive.
- Text strings (quoted in ‘’ or “”) are CASE SENSITIVE
- Keywords are highlighted in SQL Developers. Don’t use them as table/column names
- Quote values of these types: CHAR, VARCHAR2, DATE, and TIMESTAMP

Create Tables (1)



```
CREATE TABLE Customer_T
        (CustomerID                  NUMBER(11,0)     NOT NULL,
        CustomerName                 VARCHAR2(25)     NOT NULL,
        CustomerAddress              VARCHAR2(30),
        CustomerCity                 VARCHAR2(20),
        CustomerState                CHAR(2),
        CustomerPostalCode           VARCHAR2(9),
CONSTRAINT Customer_PK PRIMARY KEY (CustomerID));
```

Create Table (2)



Default value

```
CREATE TABLE Order_T
        (OrderID                    NUMBER(11,0)      NOT NULL,
        OrderDate                   DATE DEFAULT SYSDATE,
        CustomerID                  NUMBER(11,0),
CONSTRAINT Order_PK PRIMARY KEY (OrderID),
CONSTRAINT Order_FK FOREIGN KEY (CustomerID) REFERENCES Customer_T (CustomerID));
```

Table Constraint Name

Table Constraint Attribute

Table Constraint Type

Insert Data (1)

- Add a new customer with data values for **all** attributes

  INSERT INTO Customer_T **VALUES** (100, 'John Smith', '123 A Street', 'Coralville', 'IA', '52241');

  Note: 1. The data values must be in the same order as the columns in the table
  2. Put ' ' around data values of CHAR, VARCHAR2, and **DATE**

- Add a new customer with data values for **some** (but not all) attributes

  INSERT INTO Customer_T VALUES (101, ‘Matt Damon', NULL, 'Iowa City', 'IA', NULL);

  INSERT INTO Customer_T (CustomerID, CustomerName, CustomerState) VALUES (102, 'Andy Boyd', 'IL');

  Note: Those attributes without input values will be left null

Insert Data (2)

- Insert a new order with valid CustomerID

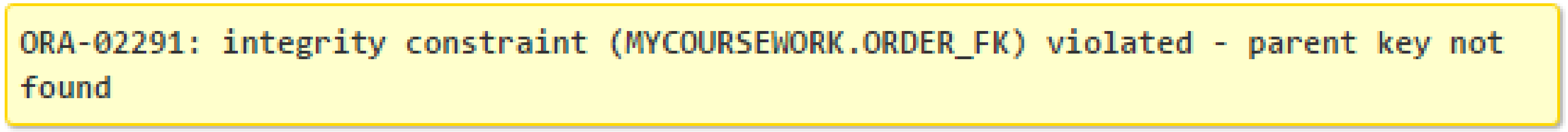
  INSERT INTO Order_T VALUES (12001, '10/16/2015', 100);

  INSERT INTO Order_T VALUES (12002, TO_DATE('2015-12-20', 'YYYY-MM-DD'), 102);

  Note: TO_DATE function converts a string to a date with the given format mask

- Insert a new order with invalid CustomerID

  INSERT INTO Order_T VALUES (12003, TO_DATE('2015-112-18', 'YYYY-MM-DD'), 99);



```
ORA-02291: integrity constraint (MYCOURSEWORK.ORDER_FK) violated - parent key not
found
```

Update/Delete Data

- Update the order date of an order

```
UPDATE Order_T
SET OrderDate = '12/28/2015'
WHERE OrderID =  12002;
```

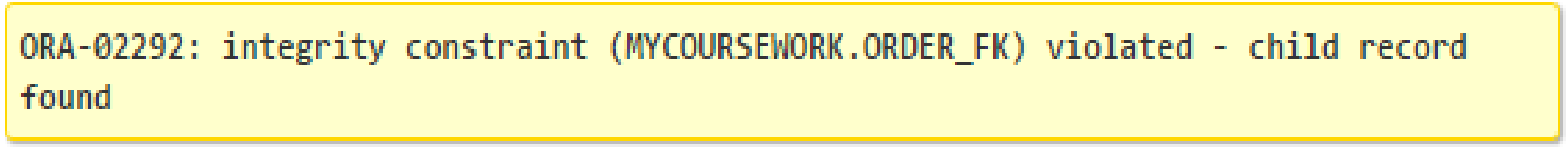
Note: WHERE clause is included to define the condition(s) for row selection

- Delete rows from a table

```
DELETE FROM Customer_T WHERE CustomerID = 101;          -- delete one row
DELETE FROM Customer_T;                                 -- delete all rows
```

Note: Deletion must be done very carefully!

What will happen if we delete CustomerID=100?



```
ORA-02292: integrity constraint (MYCOURSEWORK.ORDER_FK) violated - child record
found
```

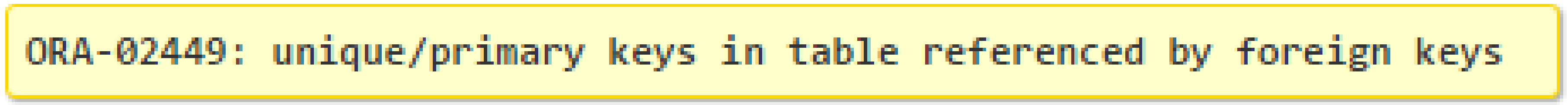
Drop Tables

- Remove a table from a database schema

  DROP TABLE Customer_T;

  DROP TABLE Order_T;

Give it a try… ☺



```
ORA-02449: unique/primary keys in table referenced by foreign keys
```

The order matters – dependent tables first!

SQL Query Structure

- **Syntax**

```
SELECT      <list of column expressions>
FROM        <list of tables and join operations>
WHERE       <list of logical expressions for rows>
GROUP BY    <list of grouping columns>
HAVING      <list of logical expressions for groups>
ORDER BY    <list of sorting specifications>
```

- **Expression**

  Combination of columns, constants, operators, and functions

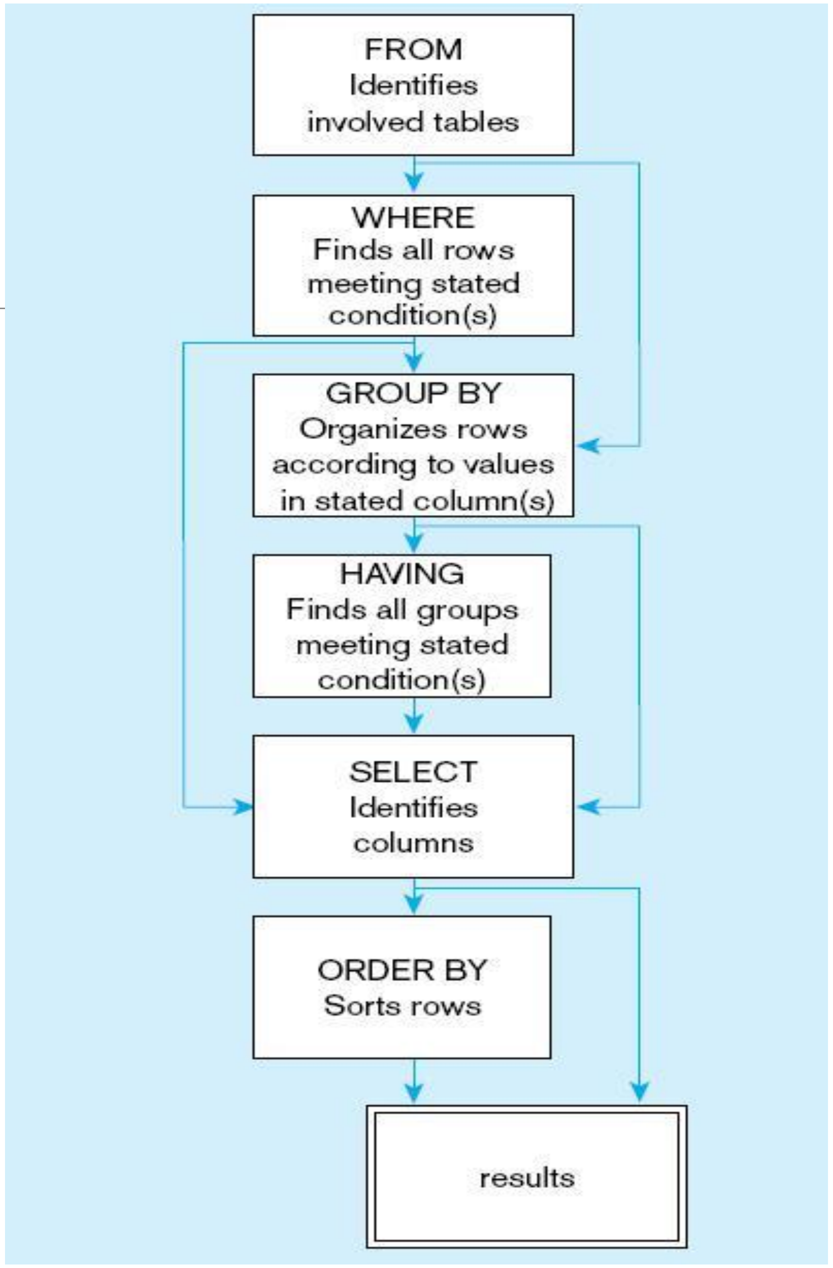
- **Input**

  A number of tables (in the FROM clause)

- **Output**

  A new temporary table (**derived**)

Table(s) in, table out



FROM Identifies involved tables
WHERE Finds all rows meeting stated condition(s)
GROUP BY Organizes rows according to values in stated column(s)
HAVING Finds all groups meeting stated condition(s)
SELECT Identifies columns
ORDER BY Sorts rows
results

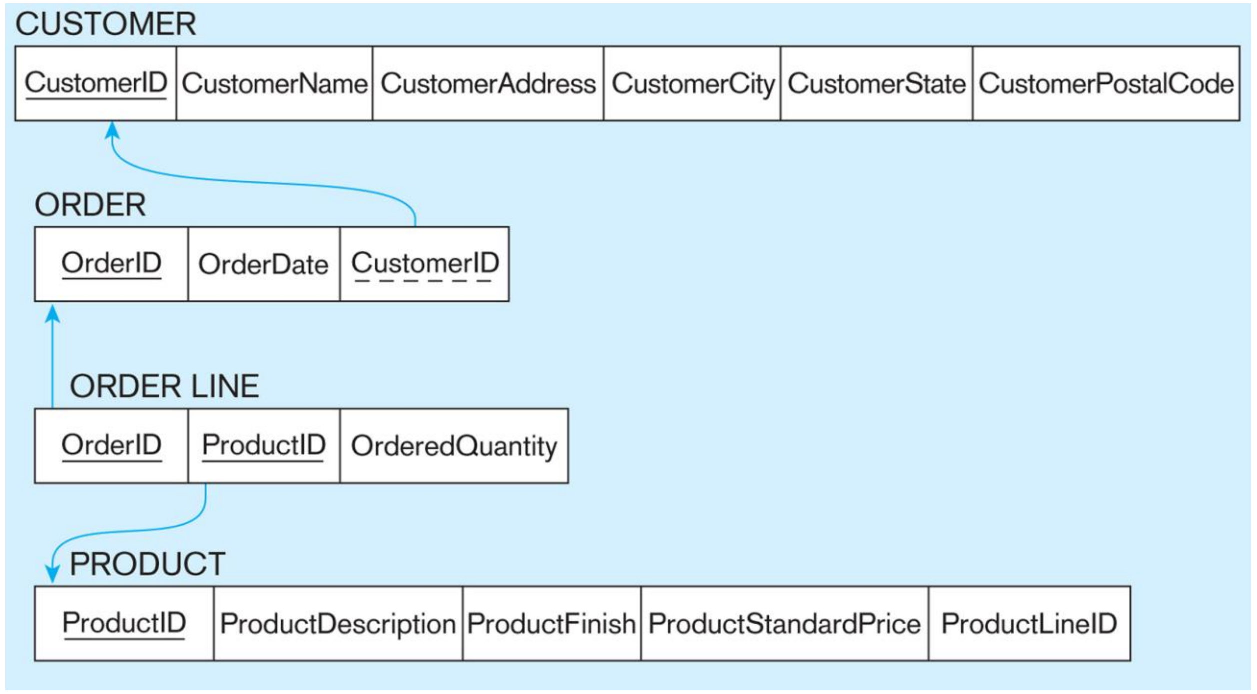
FROM Clause

- List the tables as input and Join them if needed
- Choose the right tables in a simple query

SELECT …. FROM <Table Name>;

SELECT * FROM Customer_T;

Show all the data in the Customer_T table.



SELECT Clause

Select what **COLUMNS to include** in the output

What to select	Meaning	Example
SELECT *	Select all the columns	`SELECT * FROM Order_T;`
SELECT DISTINCT Col_Name	Select distinct values in a column, remove duplicates	`SELECT DISTINCT OrderID FROM OrderLine_T;`
SELECT Col_Name AS Alias	Select a column and rename it	`SELECT CustomerName AS Name, CustomerAddress AS Address FROM Customer_T;`
SELECT expressions	Generate a new column using existing ones	`SELECT ProductID, ProductStandardPrice*1.1 AS Plus10Percent FROM Product_T;`
SELECT agg_function(COLUMN)	Calculate an aggregate function over a certain column	`SELECT AVG(ProductStandardPrice) FROM Product_T;`

SELECT Clause (1)

`Using the asterisk (*)` ← To display all columns from all items in the FROM clause

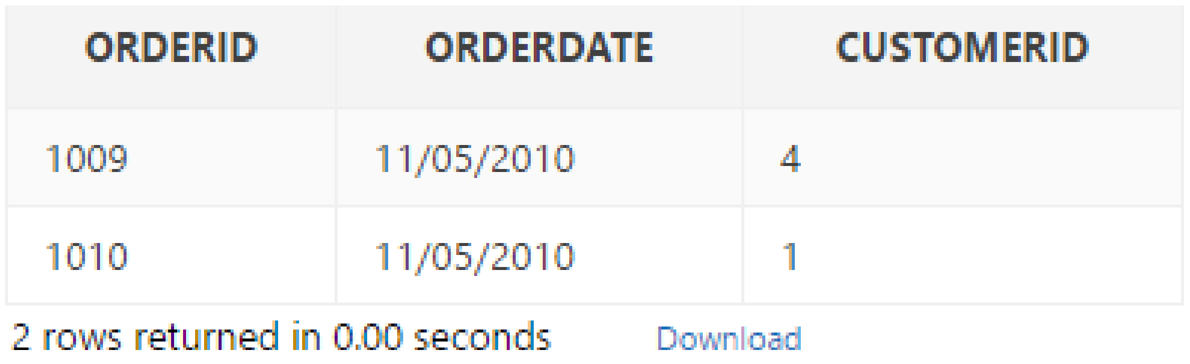
Query: Display all columns for all orders that have been placed after 10/31/2010

```
SELECT *
FROM Order_T
WHERE OrderDate > '10/31/2010';
```

← Use comparison operator with DATE

Note: the data is enclosed in single quotes

Output Table:



ORDERID	ORDERDATE	CUSTOMERID
1009	11/05/2010	4
1010	11/05/2010	1

2 rows returned in 0.00 seconds Download

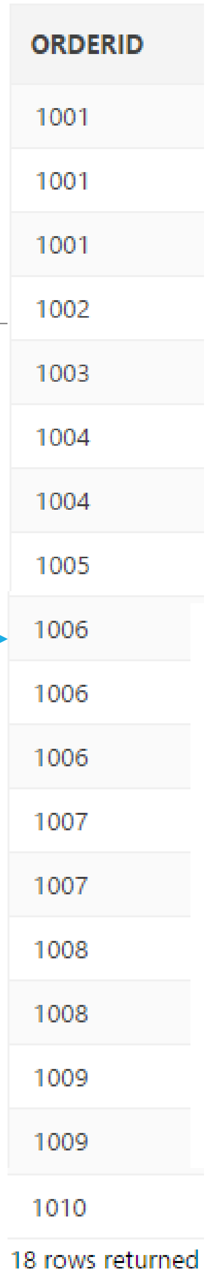
SELECT Clause (2)

Using DISTINCT ← To avoid displaying duplicate rows in the result

Query: Display (distinct) order IDs included in the OrderLine table

```
SELECT OrderID
FROM OrderLine_T;
```

Results with many duplicate rows



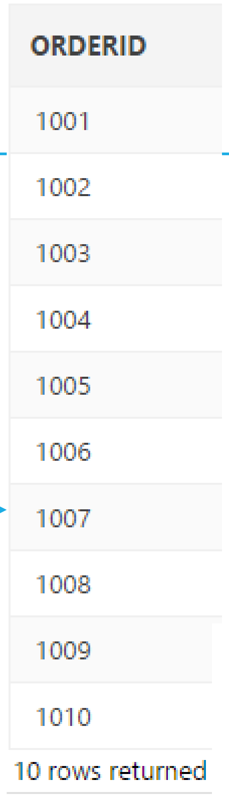
ORDERID
1001
1001
1001
1002
1003
1004
1004
1005
1006
1006
1006
1007
1007
1008
1008
1009
1009
1010

18 rows returned

Add the keyword DISTINCT

```
SELECT DISTINCT OrderID
FROM OrderLine_T;
```

Results with distinct order IDs



ORDERID
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010

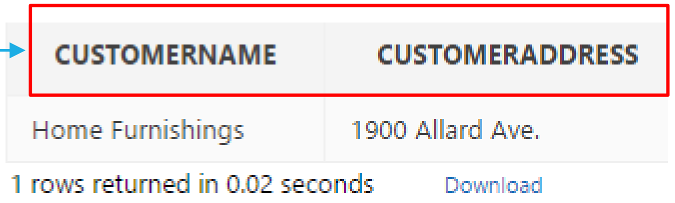
10 rows returned

SELECT Clause (3)

Using alias (AS clause) ← To redefine a column heading for display and convenience

Query: Show the address of the customer named “Home Furnishing”

```
SELECT CustomerName, CustomerAddress
FROM Customer_T
WHERE CustomerName = 'Home Furnishings';
```

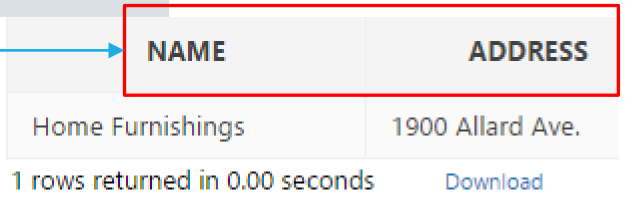


CUSTOMERNAME	CUSTOMERADDRESS
Home Furnishings	1900 Allard Ave.

1 rows returned in 0.02 seconds Download

Use alias to improve readability (and simplification)

```
SELECT CustomerName AS Name, CustomerAddress AS Address
FROM Customer_T
WHERE CustomerName = 'Home Furnishings';
```



NAME	ADDRESS
Home Furnishings	1900 Allard Ave.

1 rows returned in 0.00 seconds Download

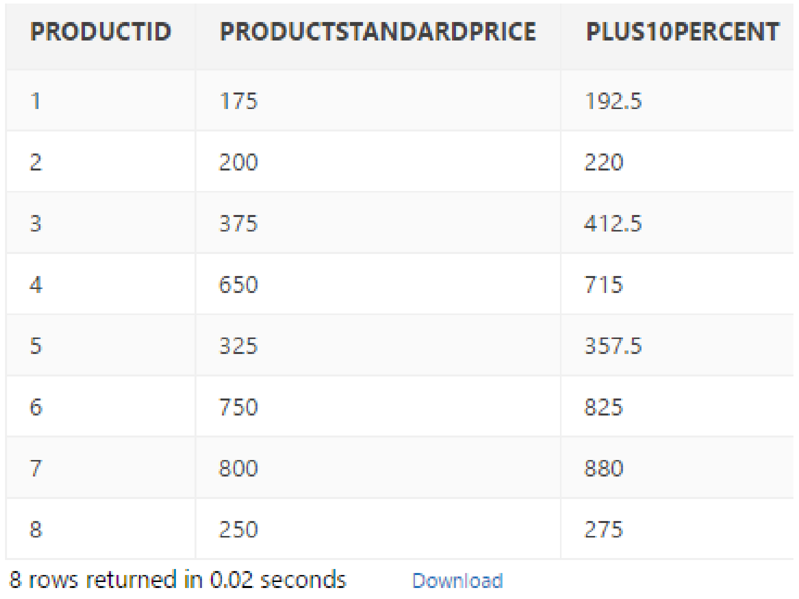
SELECT Clause (4)

Using expression and generating new columns

Query: List the standard price and the new price of 10% increase for every product

```
SELECT ProductID, ProductStandardPrice, ProductStandardPrice*1.1 AS Plus10Percent
FROM Product_T;
```

Output Table:



PRODUCTID	PRODUCTSTANDARDPRICE	PLUS10PERCENT
1	175	192.5
2	200	220
3	375	412.5
4	650	715
5	325	357.5
6	750	825
7	800	880
8	250	275

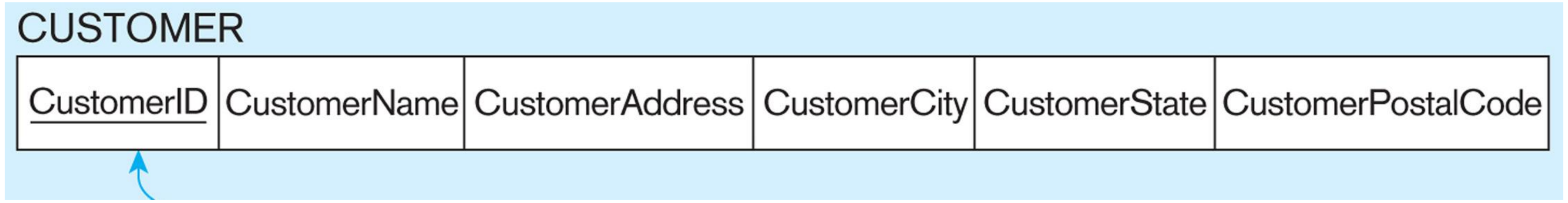
8 rows returned in 0.02 seconds Download

Exercise

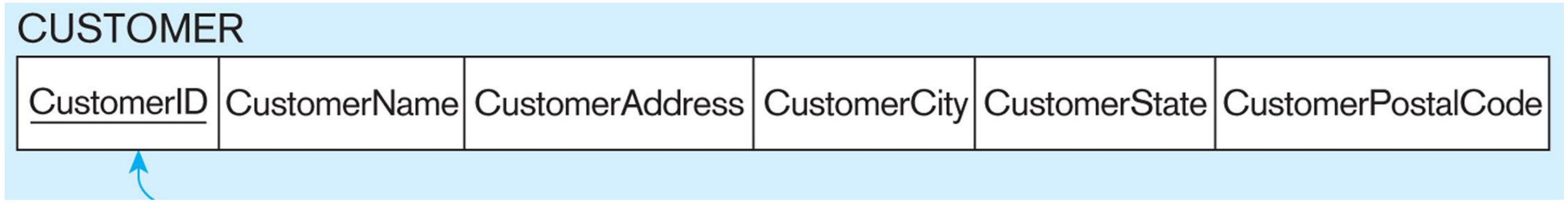
1. Write a SQL query to show the city and state combinations of all the customers. Remove duplicates in the output.

2. Write a SQL query to show each customer's ID, Name and full address as a single column (New column name FullAddress).

(hint: use the "||" operator to concatenate string values. E.g., 'ABC' || 'DEF' = 'ABCDEF')



Exercise



1. Write a SQL query to show the city and state combinations of all the customers. Remove duplicates in the output.

SELECT DISTINCT CustomerCity, CustomerState

FROM Customer_T.

2. Write a SQL query to show each customer's ID, Name and full address as a single column (New column name FullAddress).

SELECT CustomerID, CustomerName, (CustomerAddress ||' '|| CustomerCity ||' '|| CustomerState ||' '|| CustomerPostalCode) as FullAddress

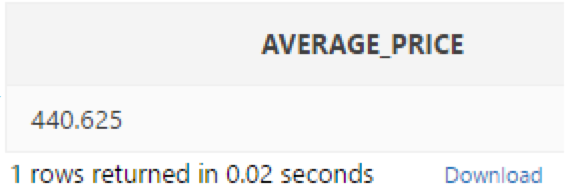
FROM Customer_T.

SELECT Clause (5)

Using aggregate functions ← To perform basic statistical analysis

Query: What is the average standard price for all products?

```
SELECT AVG(ProductStandardPrice) AS Average_Price
FROM Product_T;
```

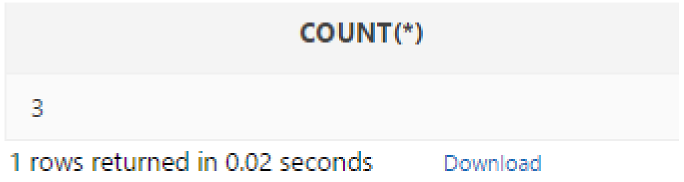


AVERAGE_PRICE
440.625

1 rows returned in 0.02 seconds Download

Query: How many different products were ordered on OrderID 1006?

```
SELECT Count(*)
FROM OrderLine_T
WHERE OrderID = 1006;
```



COUNT(*)
3

1 rows returned in 0.02 seconds Download

Other Commonly used aggregate functions: MIN, MAX, SUM, etc.

Note: using any of these aggregate functions will give a one-row answer

SELECT Clause (6)

More about COUNT function

```
SELECT COUNT(*),COUNT(CustomerID),COUNT(DISTINCT CustomerID)
FROM Order_T;
```

OrderTable

OrderID	OrderDate	CustomerID	OrderDue
12001	10/10/2014	100	100.00
12002	10/12/2014	100	200.00
12003	10/21/2014	101	NULL
12004	10/31/2013	NULL	120.00

1. COUNT(*): count all rows including NULL values),
2. COUNT(CustomerID): ignore NULL values
3. COUNT(DISTINCT CustomerID): count different customers and ignore NULL values)

Output Table (1 row)

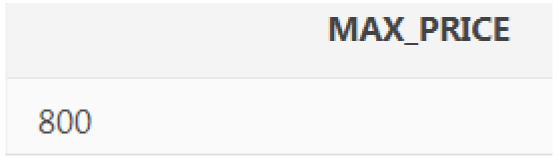
COUNT(*)	COUNT(CustomerID)	COUNT(DISTINCT CustomerID)
4	3	2

SELECT Clause (7)

More about aggregate functions ← An aggregate function gives a one-row answer

Query: What is the highest standard price for all products?

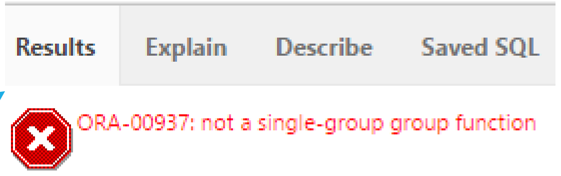
```
SELECT MAX(ProductStandardPrice) AS Max_Price
FROM Product_T;
```



```
MAX_PRICE
800
```

Query: Show the product ID and standard price of the most
expensive product.

```
SELECT ProductID, MAX(ProductStandardPrice)
FROM Product_T;
```



```
ORA-00937: not a single-group group function
```

You can NOT mix a column value (set value) with an aggregate (one-row value) in the SELECT clause!

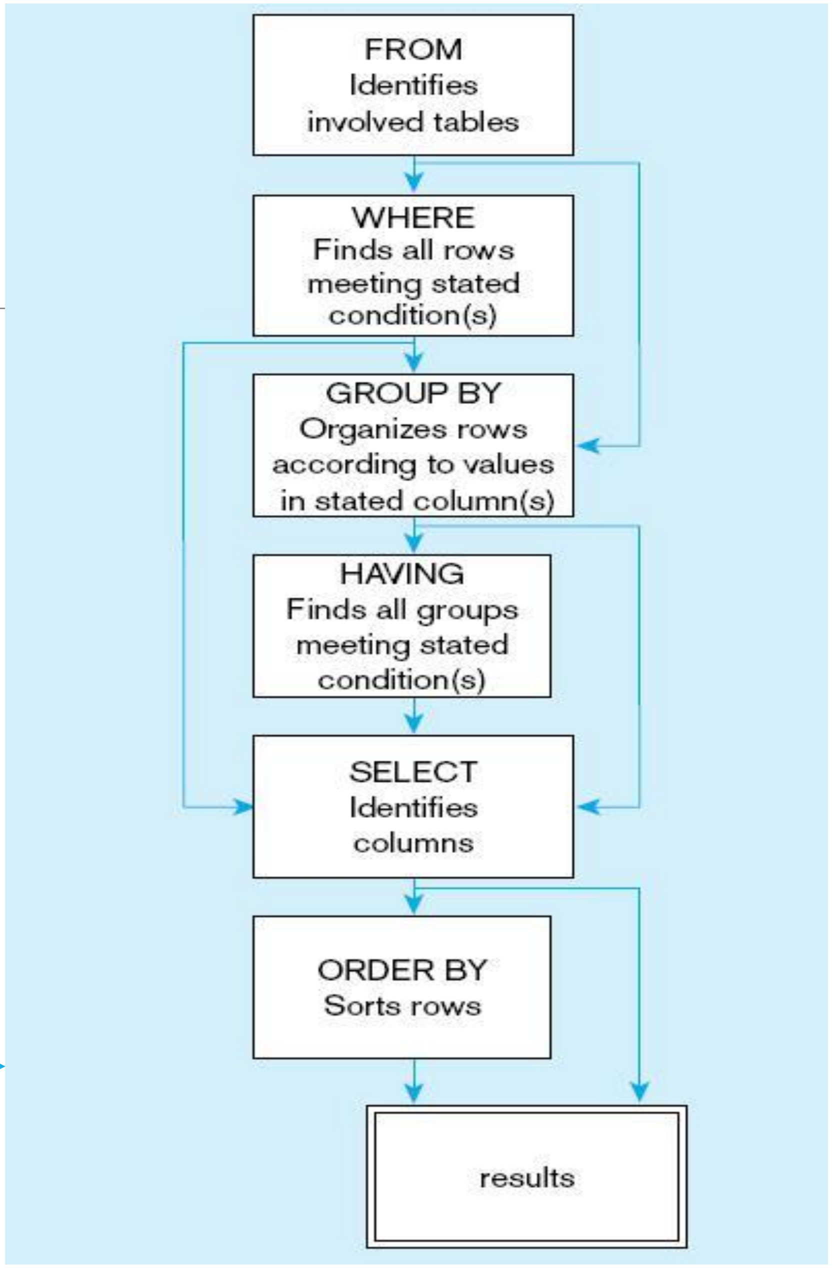
SQL Query Structure

- **Syntax**

  SELECT <list of column expressions>
  FROM <list of tables and join operations>
  WHERE <list of logical expressions for rows>
  GROUP BY <list of grouping columns>
  HAVING <list of logical expressions for groups>
  ORDER BY <list of sorting specifications>

Syntax oder

Processing order


FROM
Identifies involved tables
WHERE
Finds all rows meeting stated condition(s)
GROUP BY
Organizes rows according to values in stated column(s)
HAVING
Finds all groups meeting stated condition(s)
SELECT
Identifies columns
ORDER BY
Sorts rows
results

WHERE Clause (1)

- **Syntax**

  SELECT <list of column expressions>
  FROM <list of tables and join operations>
  WHERE <list of logical expressions for rows>
  GROUP BY <list of grouping columns>
  HAVING <list of logical expressions for groups>
  ORDER BY <list of sorting specifications>

Find all rows meeting stated condition(s)

- **Expression**
  - LIKE paired with wildcard characters (%, _)
  - Equality comparison operator (=, >, >=, <, <=, <>, !=)
  - Ranges for qualification (BETWEEN…AND, NOT BETWEEN…AND)
  - Boolean operators (AND, OR, NOT)
  - Set operators (IN, NOT IN)
  - Null values (IS NULL, IS NOT NULL)

The value of each expression is TRUE or FALSE.

SQL tests each row with the entire WHERE clause and selects those returning TRUE

WHERE Clause (2)

Using LIKE with wildcards

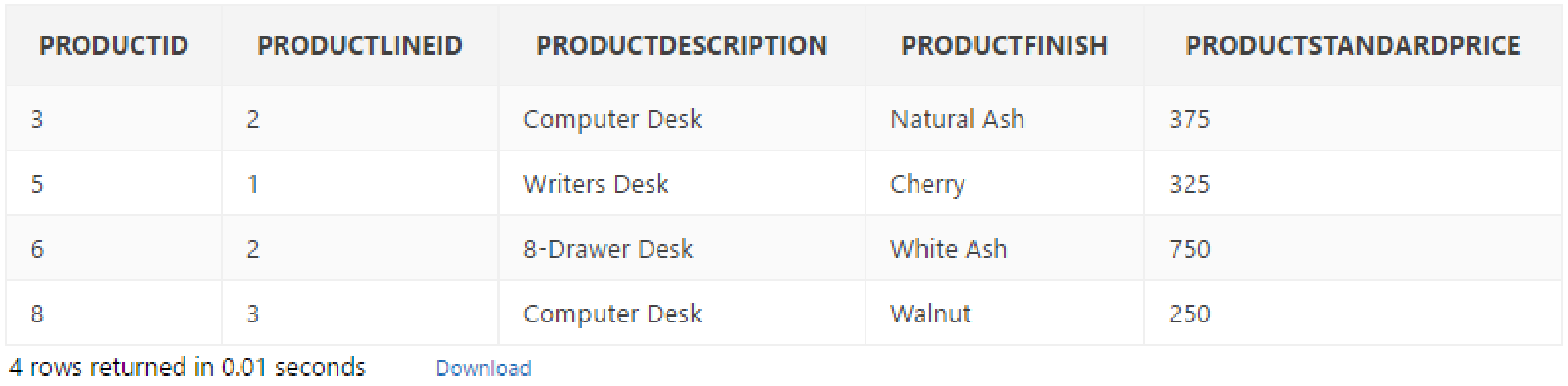
To select a batch of desired match or when an exact match is not possible

Query: Find all different types of desks carried by PVFC

```
SELECT *
FROM Product_T
WHERE ProductDescription LIKE '%Desk%';
```

(%): any collection of characters
(_): exactly one character (e.g., `'_-Desk%'`)
Note: it is case sensitive

Output Table:



PRODUCTID	PRODUCTLINEID	PRODUCTDESCRIPTION	PRODUCTFINISH	PRODUCTSTANDARDPRICE
3	2	Computer Desk	Natural Ash	375
5	1	Writers Desk	Cherry	325
6	2	8-Drawer Desk	White Ash	750
8	3	Computer Desk	Walnut	250

4 rows returned in 0.01 seconds Download

WHERE Clause (3)

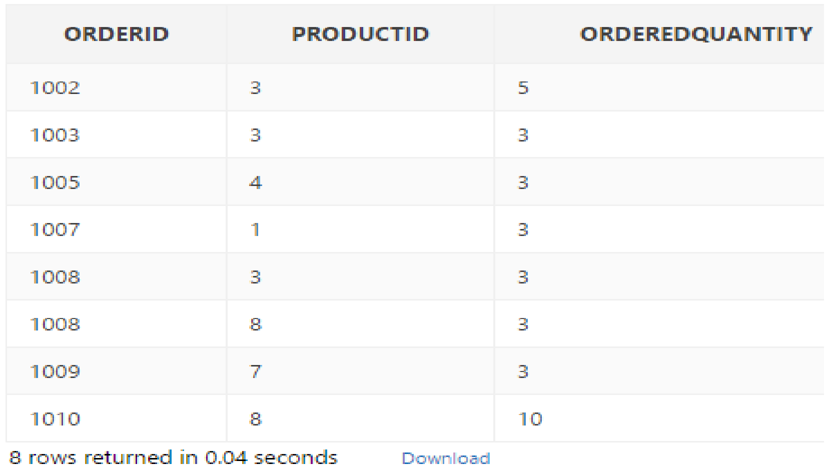
Using comparison operators ← **with numeric data**

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

Query: Find the orders with a quantity of more than 2

```
SELECT *
FROM OrderLine_T
WHERE OrderedQuantity > 2;
```

Output Table:



ORDERID	PRODUCTID	ORDEREDQUANTITY
1002	3	5
1003	3	3
1005	4	3
1007	1	3
1008	3	3
1008	8	3
1009	7	3
1010	8	10

8 rows returned in 0.04 seconds Download

```
SELECT *
FROM Order_T
WHERE OrderDate > '10/31/2010';
```

WHERE Clause (4)

Using comparison operators

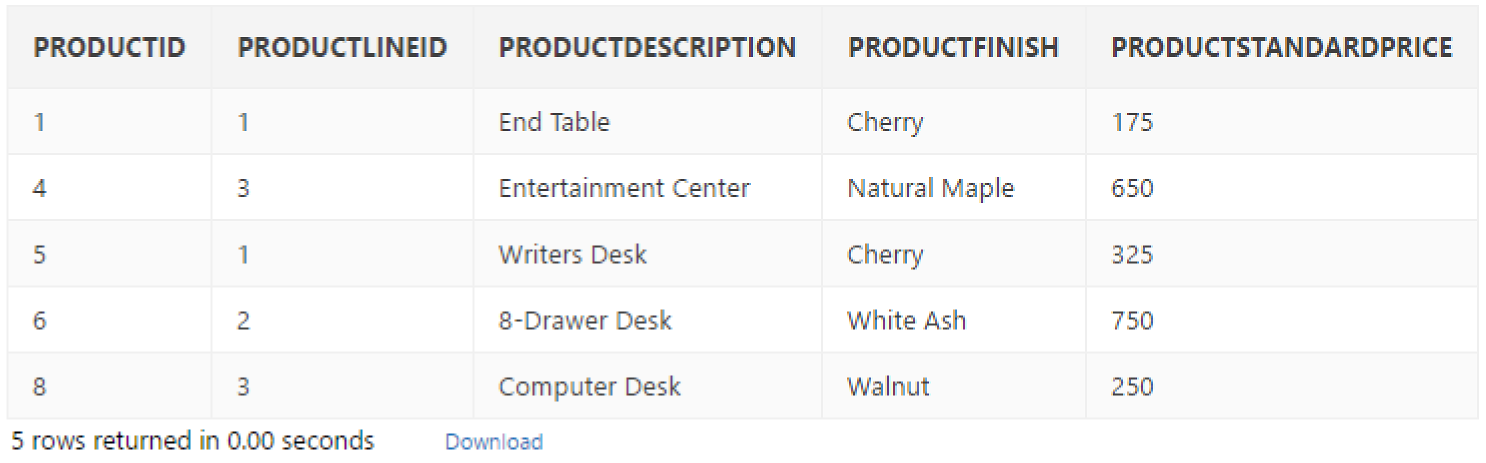
with character data and dates as well

Query: What furniture does PVFC carry that is not made of natural ash?

```
SELECT *
FROM Product_T
WHERE ProductFinish != 'Natural Ash';
```

Note: exact match

Output Table:



PRODUCTID	PRODUCTLINEID	PRODUCTDESCRIPTION	PRODUCTFINISH	PRODUCTSTANDARDPRICE
1	1	End Table	Cherry	175
4	3	Entertainment Center	Natural Maple	650
5	1	Writers Desk	Cherry	325
6	2	8-Drawer Desk	White Ash	750
8	3	Computer Desk	Walnut	250

5 rows returned in 0.00 seconds Download

WHERE Clause (5)

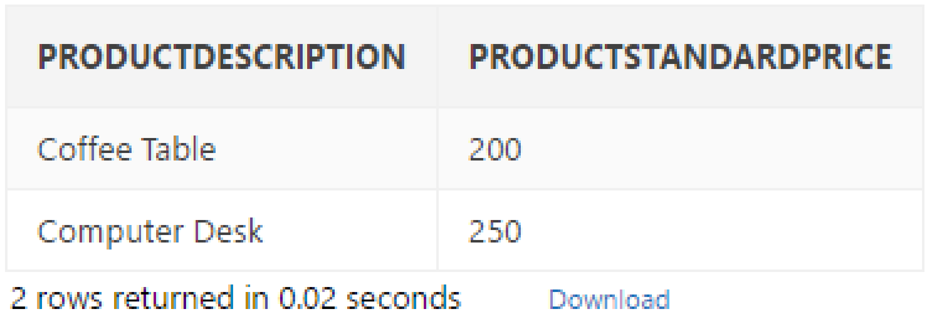
Using ranges

Establish a range of values for qualification
[BTWEEN…AND, NOT BETWEEN…AND]

Query: Which products carried by PVFC have a standard price between $200 and $300?

```
SELECT ProductDescription, ProductStandardPrice
FROM Product_T
WHERE ProductStandardPrice BETWEEN 200 AND 300;
```

Output Table:



PRODUCTDESCRIPTION	PRODUCTSTANDARDPRICE
Coffee Table	200
Computer Desk	250

2 rows returned in 0.02 seconds Download

Note: both 200 and 300 are included in the range

WHERE Clause (6)

What is a NULL value?

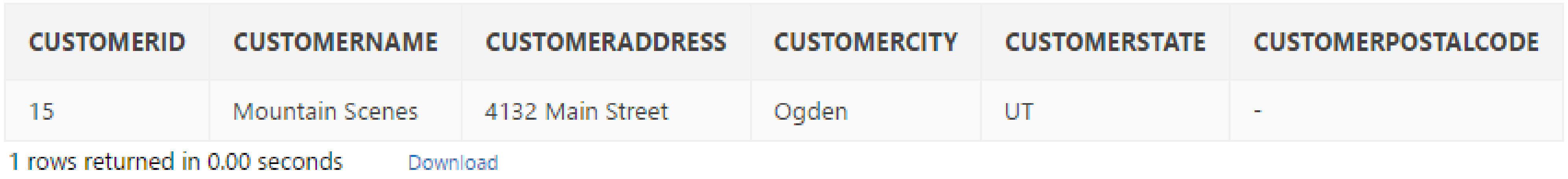
Using Null values (IS NULL and IS NOT NULL)

to find rows with or without missing values for a column/attribute

Query: Display all customers whose postal code is unknown/missing

```
SELECT *
FROM Customer_T
WHERE CustomerPostalCode IS NULL;
```

Output Table:



CUSTOMERID	CUSTOMERNAME	CUSTOMERADDRESS	CUSTOMERCITY	CUSTOMERSTATE	CUSTOMERPOSTALCODE
15	Mountain Scenes	4132 Main Street	Ogden	UT	-

1 rows returned in 0.00 seconds Download

WHERE Clause (7)

`Using Boolean operators` ← Refine the conditions and make SQL queries more complex and more powerful

AND	Join two or more conditions and returns results only when all conditions are true [Intersect]
OR	Join two or more conditions and returns results when any conditions are true [Union]
NOT	Negate an expression

Note: if they are used together, NOT is evaluated first, then AND, then OR, i.e., NOT→AND→OR.
To reduce confusion, use parentheses.

WHERE Clause (8)

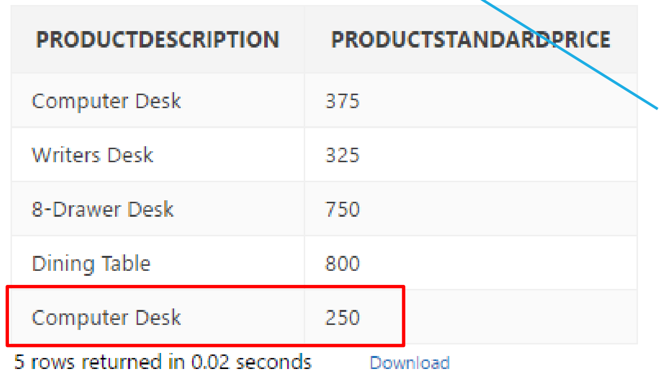
Boolean operation precedence

NOT → AND → OR

Query: List product description and standard price for all desks, and all tables that cost more than $300

```
SELECT ProductDescription, ProductStandardPrice
FROM Product_T
WHERE ProductDescription LIKE '%Desk'
   OR (ProductDescription LIKE '%Table'
  AND ProductStandardPrice > 300);
```

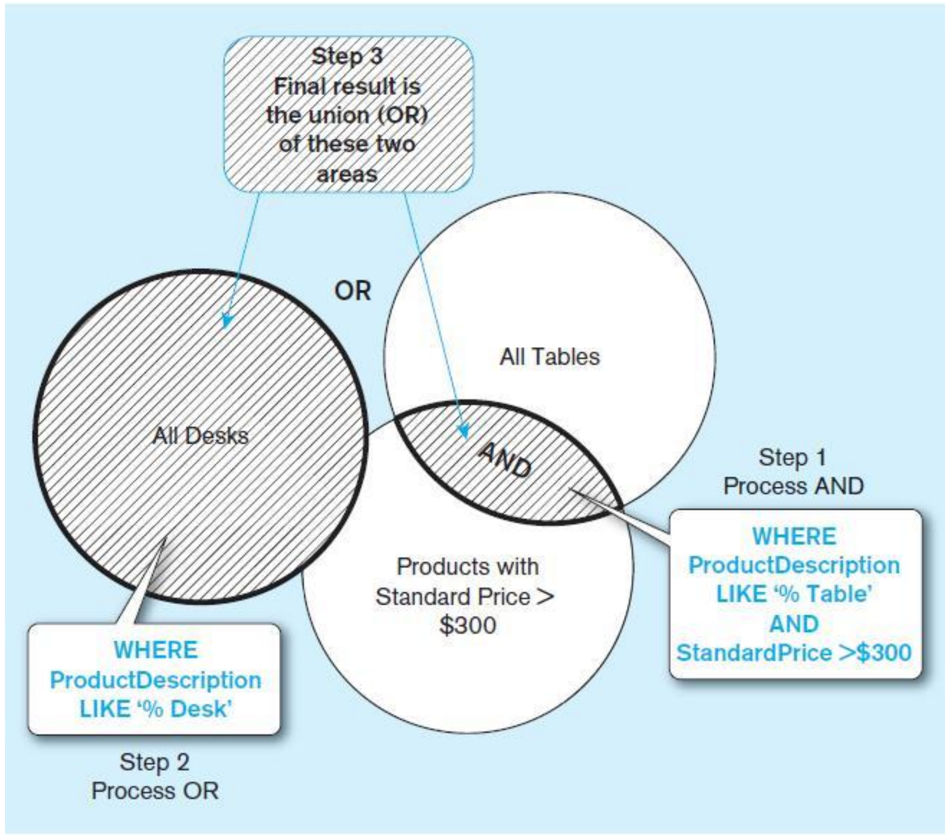
Output Table:



PRODUCTDESCRIPTION	PRODUCTSTANDARDPRICE
Computer Desk	375
Writers Desk	325
8-Drawer Desk	750
Dining Table	800
Computer Desk	250

5 rows returned in 0.02 seconds Download

Use parentheses even though it is redundant in this case


Step 3
Final result is the union (OR) of these two areas
OR
All Tables
All Desks
AND
Step 1
Process AND
Products with Standard Price > $300
WHERE ProductDescription LIKE '% Table' AND StandardPrice >$300
WHERE ProductDescription LIKE '% Desk'
Step 2
Process OR

WHERE Clause (9)

Boolean operation with parentheses ← **explicitly denote precedence (recommended)**

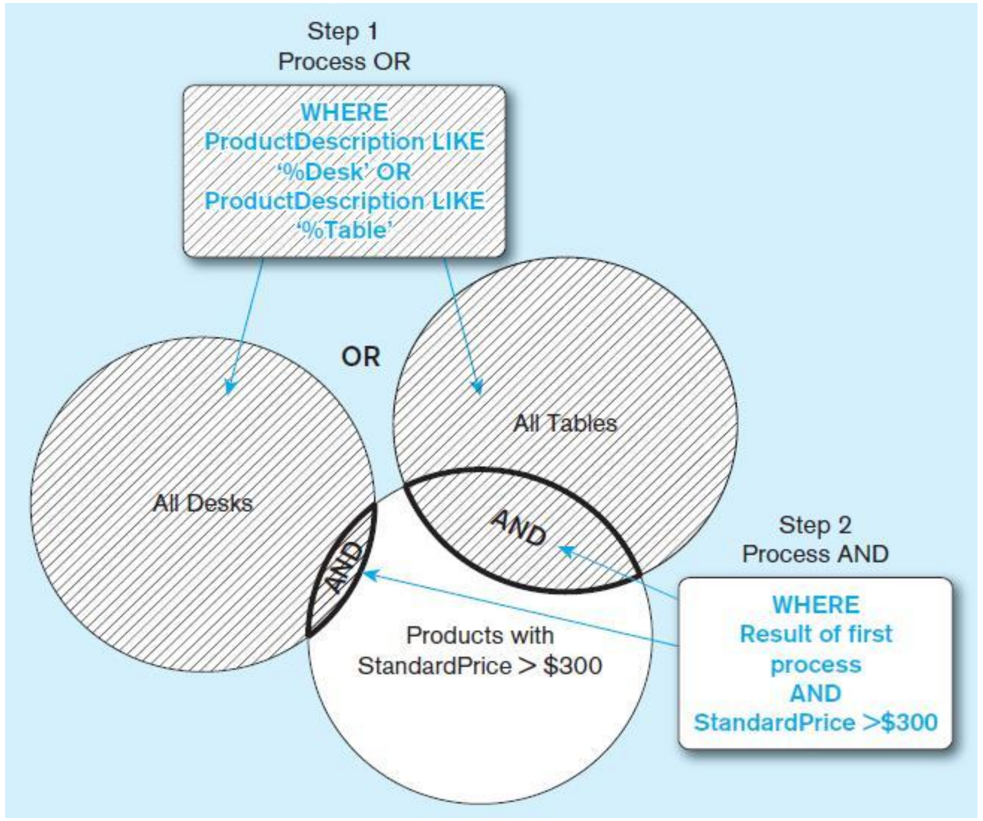
Query: List product description and standard price for all (desks and tables) that cost more than $300

```
SELECT ProductDescription, ProductStandardPrice
FROM Product_T
WHERE (ProductDescription LIKE '%Desk'
   OR ProductDescription LIKE '%Table')
  AND ProductStandardPrice > 300;
```

Output Table:

PRODUCTDESCRIPTION	PRODUCTSTANDARDPRICE
Computer Desk	375
Writers Desk	325
8-Drawer Desk	750
Dining Table	800

4 rows returned in 0.02 seconds   Download


Step 1
Process OR
WHERE ProductDescription LIKE '%Desk' OR ProductDescription LIKE '%Table'
OR
All Desks
All Tables
AND
AND
Step 2
Process AND
Products with StandardPrice > $300
WHERE Result of first process AND StandardPrice >$300

WHERE Clause (10)

Using set operators (IN and NOT IN)

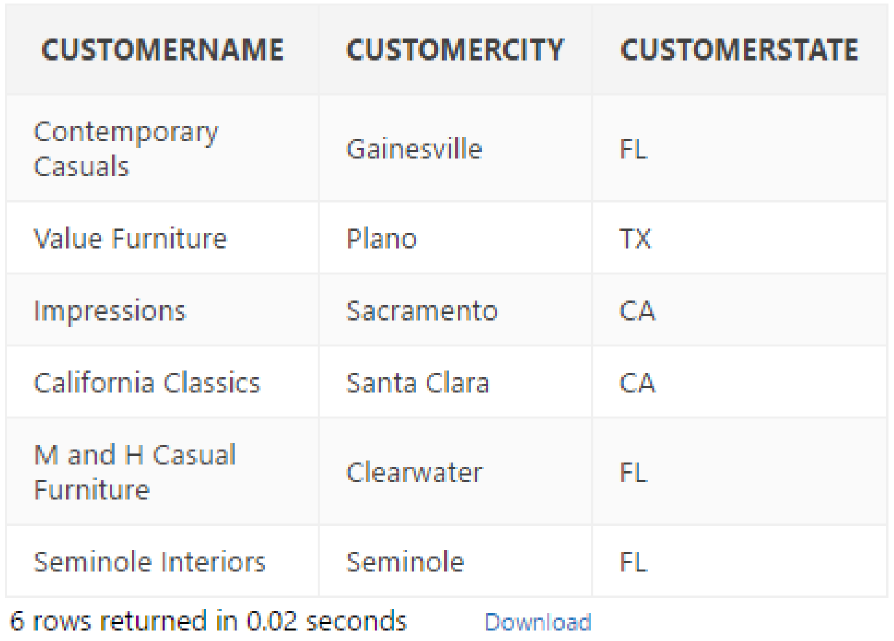
Match a list of values (very useful)

Query: List all customers who live in Florida, Texas, or California

```
SELECT CustomerName,CustomerCity,CustomerState
FROM Customer_T
WHERE CustomerState IN ('FL','TX','CA');
```

Note: the list (set of values) inside the parentheses after IN can be literal or a SELECT statement with a single result column (subquery)

Output Table:



CUSTOMERNAME	CUSTOMERCITY	CUSTOMERSTATE
Contemporary Casuals	Gainesville	FL
Value Furniture	Plano	TX
Impressions	Sacramento	CA
California Classics	Santa Clara	CA
M and H Casual Furniture	Clearwater	FL
Seminole Interiors	Seminole	FL

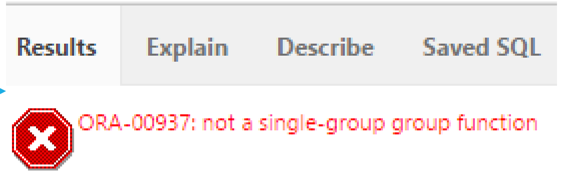
6 rows returned in 0.02 seconds Download

WHERE Clause (11)

This question is solved by using a subquery (discussed later)

Query: Show the product ID and standard price of the most expensive product.

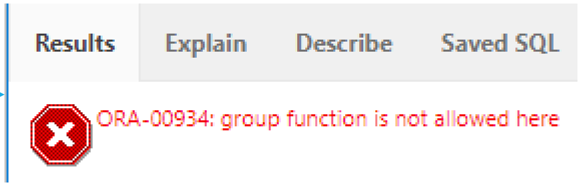
```
SELECT ProductID, MAX(ProductStandardPrice)
FROM Product_T;
```



ORA-00937: not a single-group group function

You can NOT mix a column value (set value) with an aggregate (one-row value) in the SELECT clause!

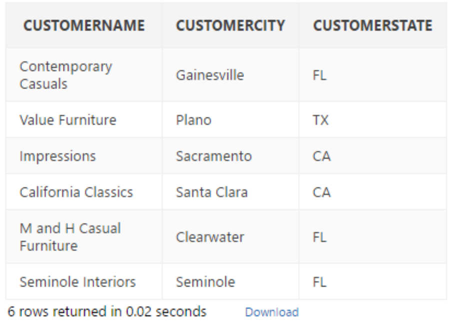
```
SELECT ProductID, MAX(ProductStandardPrice)
FROM Product_T
WHERE ProductStandardPrice = MAX(ProductStandardPrice);
```



ORA-00934: group function is not allowed here

Can NOT directly put an aggregate in the WHERE clause

ORDER BY Clause (1)



CUSTOMERNAME	CUSTOMERCITY	CUSTOMERSTATE
Contemporary Casuals	Gainesville	FL
Value Furniture	Plano	TX
Impressions	Sacramento	CA
California Classics	Santa Clara	CA
M and H Casual Furniture	Clearwater	FL
Seminole Interiors	Seminole	FL

6 rows returned in 0.02 seconds Download

Character data

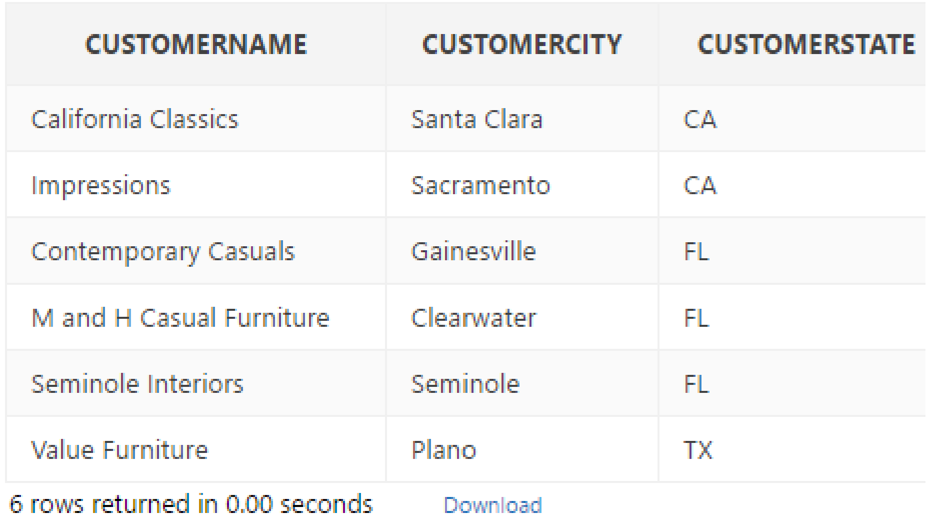
Sort results in ascending/descending order of some attribute(s)

Query: Show all customers who live in Florida, Texas, or California, and list the customers alphabetically by state and alphabetically by customer name within each state.

```
SELECT CustomerName,CustomerCity,CustomerState
FROM Customer_T
WHERE CustomerState IN ('FL','TX','CA')
ORDER BY CustomerState, CustomerName;
```

Note: the sorting order is determined by the order in which the columns are listed in the ORDER BY clause.

Output Table:



CUSTOMERNAME	CUSTOMERCITY	CUSTOMERSTATE
California Classics	Santa Clara	CA
Impressions	Sacramento	CA
Contemporary Casuals	Gainesville	FL
M and H Casual Furniture	Clearwater	FL
Seminole Interiors	Seminole	FL
Value Furniture	Plano	TX

6 rows returned in 0.00 seconds Download

ORDER BY Clause (2)

Numeric data

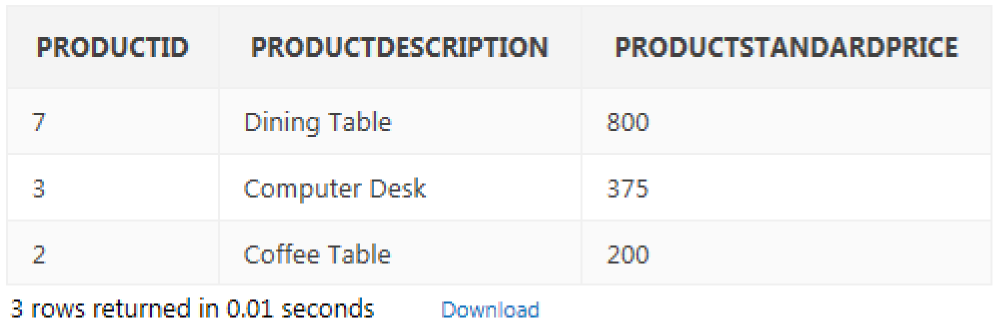
Sort results in ascending/descending order of some attribute(s)

Query: List the products with natural ash finish based on their standard prices from high to low.

```
SELECT ProductID, ProductDescription,
       ProductStandardPrice
FROM Product_T
WHERE ProductFinish = 'Natural Ash'
ORDER BY ProductStandardPrice DESC;
```

Note: To sort in descending order, place DESC after the column used to sort.

Output Table:



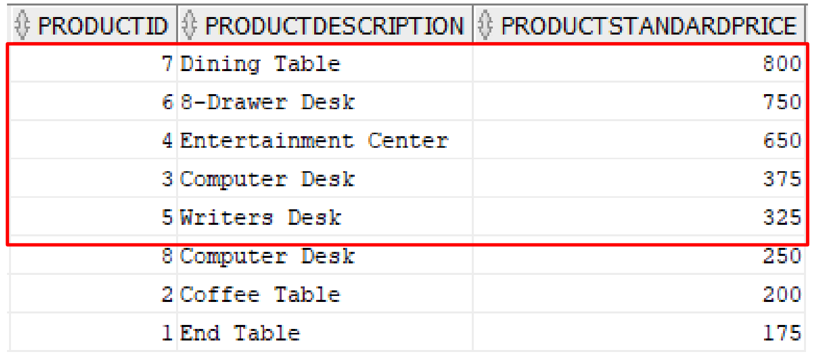
PRODUCTID	PRODUCTDESCRIPTION	PRODUCTSTANDARDPRICE
7	Dining Table	800
3	Computer Desk	375
2	Coffee Table	200

3 rows returned in 0.01 seconds   Download

Top-K Query

- Question: Find the top-5 most expensive products
- First rank the products by price from high to low, then pick the first 5 rows

```
SELECT ProductID, ProductDescription,
       ProductStandardPrice
FROM Product_T
ORDER BY ProductStandardPrice DESC;
```

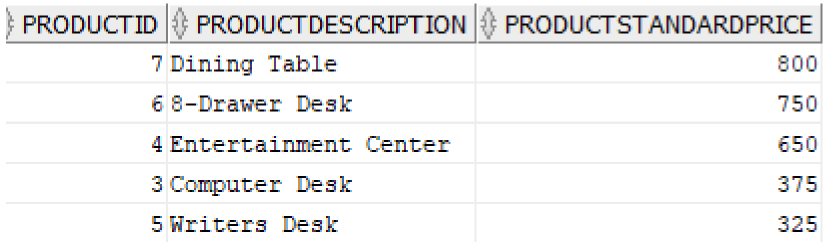


PRODUCTID	PRODUCTDESCRIPTION	PRODUCTSTANDARDPRICE
7	Dining Table	800
6	8-Drawer Desk	750
4	Entertainment Center	650
3	Computer Desk	375
5	Writers Desk	325
8	Computer Desk	250
2	Coffee Table	200
1	End Table	175

- In Oracle Database 11g, you need to use the “rownum” column (a hidden, system column)

```
SELECT * FROM
(SELECT ProductID, ProductDescription,
       ProductStandardPrice
FROM Product_T
ORDER BY ProductStandardPrice DESC)
WHERE rownum <= 5
```

Subquery



PRODUCTID	PRODUCTDESCRIPTION	PRODUCTSTANDARDPRICE
7	Dining Table	800
6	8-Drawer Desk	750
4	Entertainment Center	650
3	Computer Desk	375
5	Writers Desk	325

WHERE Clause (2)

`Using LIKE with wildcards`

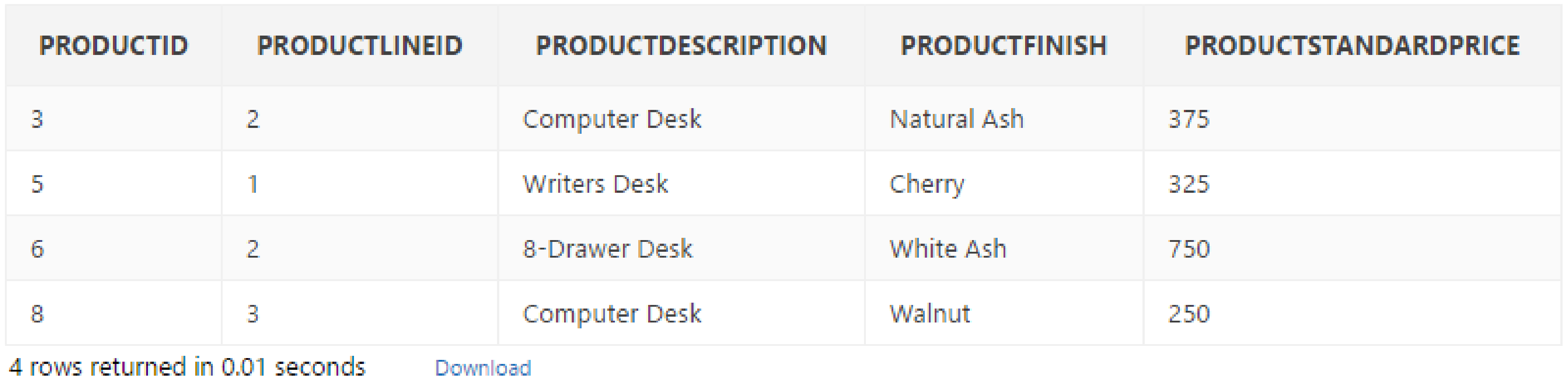
To select a batch of desired match or when an exact match is not possible

Query: Find all different types of desks carried by PVFC

```
SELECT *
FROM Product_T
WHERE ProductDescription LIKE '%Desk%';
```

(%): any collection of characters
(_): exactly one character (e.g., `'_-Desk%'`)
Note: it is case sensitive

Output Table:



PRODUCTID	PRODUCTLINEID	PRODUCTDESCRIPTION	PRODUCTFINISH	PRODUCTSTANDARDPRICE
3	2	Computer Desk	Natural Ash	375
5	1	Writers Desk	Cherry	325
6	2	8-Drawer Desk	White Ash	750
8	3	Computer Desk	Walnut	250

4 rows returned in 0.01 seconds   Download

String Matching

Difference between the following queries?

Q1

```
SELECT * FROM Product_T
WHERE ProductDescription LIKE '%Desk';
```

Find products whose description ENDS with the word "Desk"

Suffix

Q2

```
SELECT * FROM Product_T
WHERE ProductDescription LIKE 'Desk%';
```

Find products whose description BEGINS with the word "Desk"

Prefix

Q3

```
SELECT * FROM Product_T
WHERE ProductDescription LIKE '%Desk%';
```

Find products whose description with the word "Desk" anywhere

Which query will include the following products?

'Computer Desk'	'Desk'	'DESK'	'Desktop Computer'	'Computer Desk set'
Q1,Q3	Q1,Q2,Q3	None	Q2, Q3	Q3

String Matching

Difference between the following queries?

Q1
```
SELECT * FROM Product_T
WHERE ProductDescription LIKE ‘%Windows _';
```
Description must end with “Windows “ (one space) plus exactly one single character.

Q2
```
SELECT * FROM Product_T
WHERE ProductDescription LIKE ‘%Windows __';
```
Description must begin with “Windows“ (one space) plus exactly two characters.

Q3
```
SELECT * FROM Product_T
WHERE ProductDescription LIKE ‘Windows%’;
```
Description begins with “Windows” (one space).

Which query will include the following products?

‘Windows 7’	‘Microsoft Windows 10’	‘Windows XP’	‘New Windows 10 bundle’	‘Windows Vista’
Q1,Q3	Q2	Q2, Q3	None	Q3

ORDER BY Clause (1)

Sort results in ascending/descending order of some attribute(s)

```
SELECT ProductID, ProductDescription,
       ProductStandardPrice
FROM Product_T
ORDER BY ProductStandardPrice DESC;
```

```
SELECT ProductID, ProductDescription,
       ProductStandardPrice
FROM Product_T
ORDER BY ProductPrice DESC,
ProductDescription ASC;
```

Column type	ORDER BY ASC	ORDER BY DESC
Numeric	Small to large	Large to small
Char, Varchar2	A-Z	Z-A
Date, Timestamp	chronological	inverse chronological

ORDER BY COL1 ASC, COL2 DESC, …

First order the rows by COL1 in ASC order. If there are ties, order them based on COL2 in DESC order ….

GROUP BY Clause (1)

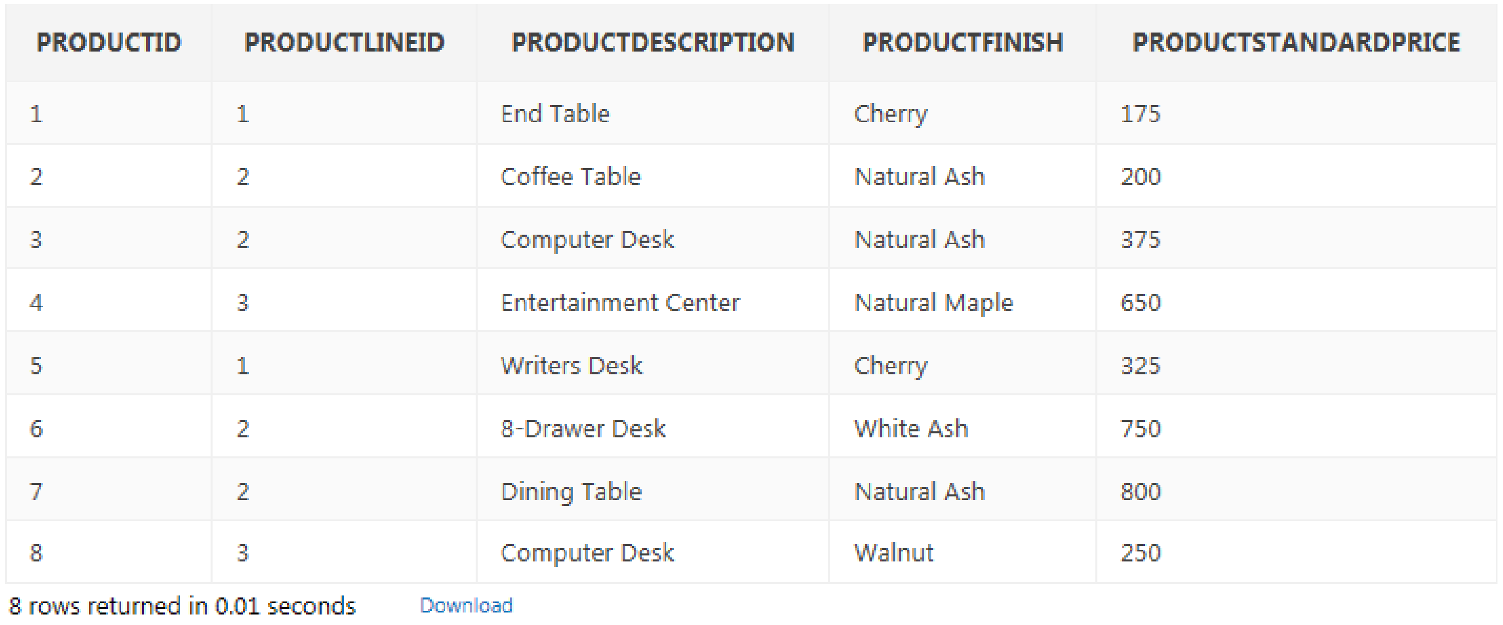
`Divide a table into subsets` ← **Usually paired with aggregate functions**

- Categorize rows into groups based on the value of the column(s)/field(s) specified
- Rows with the same value in the "GROUP BY column(s)" will be put in one group
- For EACH group, you may calculate statistics using aggregate functions
- One row for EACH group in the result table (vector aggregate)
- Only the column with a single value for each group can be included in the SELECT clause

  Rule: each column in the SELECT clause (except for aggregate functions) must be referenced in the "GROUP BY column(s)"

Scalar aggregate: the single (one-row) aggregate value returned from a SQL query without using GROUP BY clause

GROUP BY Clause (2)



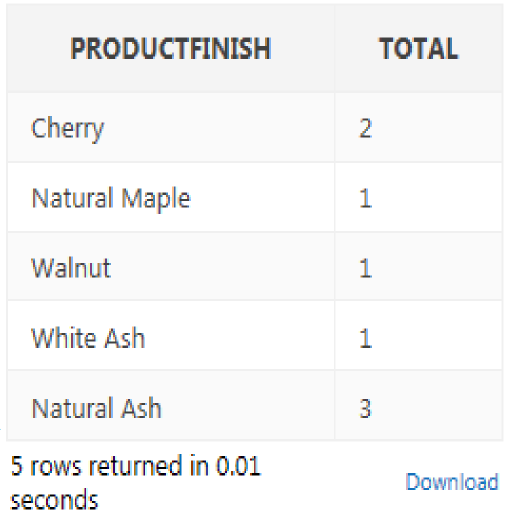
PRODUCTID	PRODUCTLINEID	PRODUCTDESCRIPTION	PRODUCTFINISH	PRODUCTSTANDARDPRICE
1	1	End Table	Cherry	175
2	2	Coffee Table	Natural Ash	200
3	2	Computer Desk	Natural Ash	375
4	3	Entertainment Center	Natural Maple	650
5	1	Writers Desk	Cherry	325
6	2	8-Drawer Desk	White Ash	750
7	2	Dining Table	Natural Ash	800
8	3	Computer Desk	Walnut	250

8 rows returned in 0.01 seconds Download

Query: How many different products for each finish type?

```
SELECT ProductFinish, COUNT(ProductID) AS Total
FROM Product_T
GROUP BY ProductFinish;
```

Output Table:



PRODUCTFINISH	TOTAL
Cherry	2
Natural Maple	1
Walnut	1
White Ash	1
Natural Ash	3

5 rows returned in 0.01 seconds Download

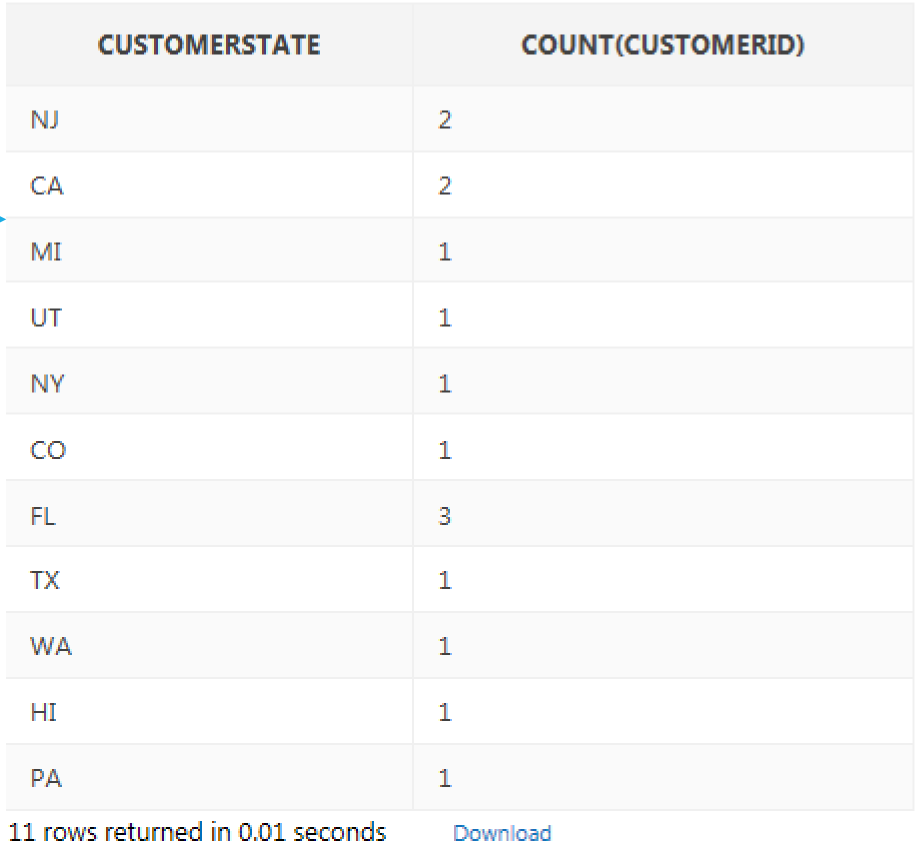
GROUP BY Clause (3)

Query: Find out the number of customers in each state

```
SELECT CustomerState,COUNT(CustomerID)
FROM Customer_T
GROUP BY CustomerState;
```

Note: Only the column/field referenced in the GROUP BY clause and the aggregate function can be included in the SELECT clause

Output Table:



CUSTOMERSTATE	COUNT(CUSTOMERID)
NJ	2
CA	2
MI	1
UT	1
NY	1
CO	1
FL	3
TX	1
WA	1
HI	1
PA	1

11 rows returned in 0.01 seconds   Download

GROUP BY multiple columns

Query: How many products with each finish are produced by each productline?

Two products are put in the same group only when they have the same ProductLineID and the same ProductFinish

```
SELECT ProductFinish, ProductLineID, COUNT(ProductID) AS Total
FROM Product_T
GROUP BY ProductFinish, ProductLineID;
```

How many groups do you expect from this query?

Number of unique combinations of (ProductFinish, ProductLineID) in the data

HAVING Clause (1)

State conditions for group selection

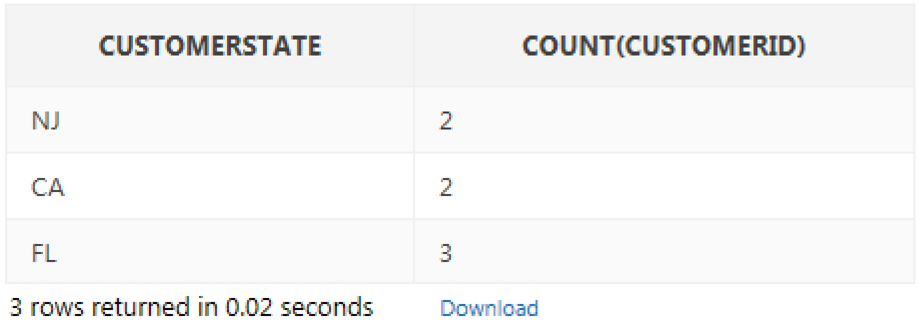
only used with a GROUP BY clause

Query: Find states with more than one customer

```
SELECT CustomerState,COUNT(CustomerID)
FROM Customer_T
GROUP BY CustomerState
HAVING COUNT(CustomerID) > 1;
```

Note: To include more than one condition, use AND, OR, and NOT just as in the WHERE clause

Output Table:



CUSTOMERSTATE	COUNT(CUSTOMERID)
NJ	2
CA	2
FL	3

3 rows returned in 0.02 seconds Download

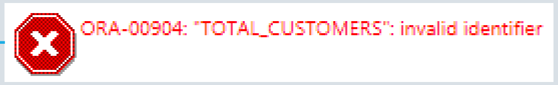
Can we replace HAVING with WHERE?

HAVING Clause (2)

Query: Find states with more than one customer and sort the states based on the number of customers from high to low

Why the alias produces an error in HAVING clause but it works well in ORDER BY clause?

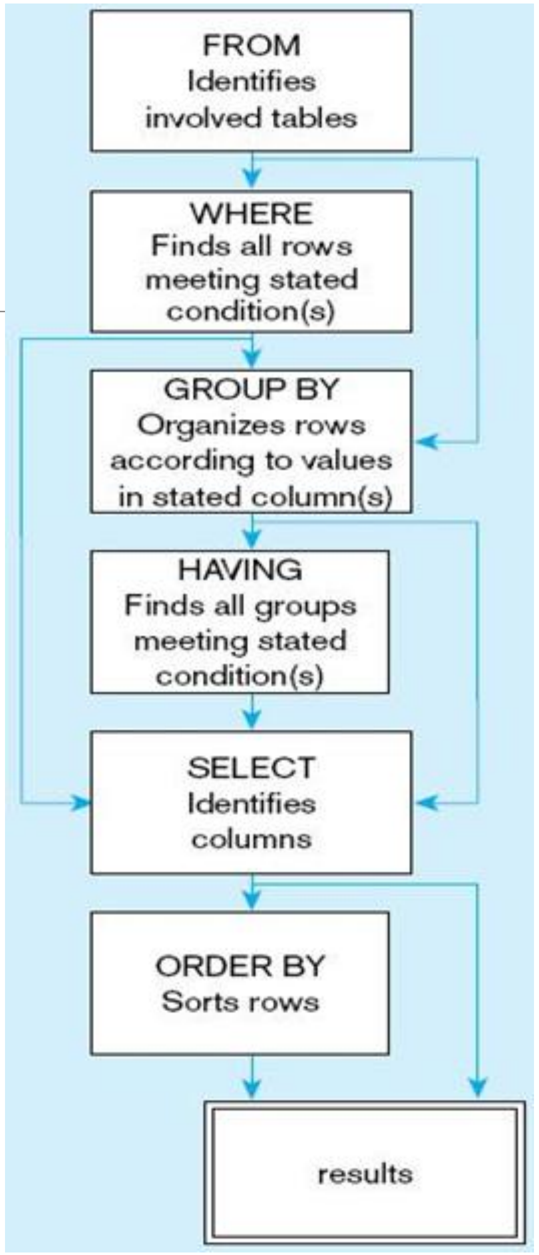
```
SELECT CustomerState AS State, COUNT(CustomerID) AS Total_Customers
FROM Customer_T
GROUP BY CustomerState
HAVING Total_Customers > 1
ORDER BY Total_Customers DESC;
```


ORA-00904: "TOTAL_CUSTOMERS": invalid identifier

```
SELECT CustomerState AS State, COUNT(CustomerID) AS Total_Customers
FROM Customer_T
GROUP BY CustomerState
HAVING COUNT(CustomerID) > 1
ORDER BY Total_Customers DESC;
```

Output Table:

STATE	TOTAL_CUSTOMERS
FL	3
NJ	2
CA	2


FROM
Identifies involved tables
WHERE
Finds all rows meeting stated condition(s)
GROUP BY
Organizes rows according to values in stated column(s)
HAVING
Finds all groups meeting stated condition(s)
SELECT
Identifies columns
ORDER BY
Sorts rows
results

HAVING Clause (3)

Use multiple test conditions in HAVING, connected by AND, OR, NOT

Similar to WHERE

Query: Find states with more than one customer but less than 4

```
SELECT CustomerState,COUNT(CustomerID)
FROM Customer_T
GROUP BY CustomerState
HAVING COUNT(CustomerID) > 1 AND COUNT(CustomerID) < 4 ;
```

```
SELECT CustomerState,COUNT(CustomerID)
FROM Customer_T
GROUP BY CustomerState
HAVING COUNT(CustomerID) BETEEN 2 AND 3;
```

DON'T use Aggregate functions in WHERE.

Common Mistakes

1. Select the columns not in GROUP BY clause

```
SELECT ProductFinish, ProductDescription, COUNT(*) FROM Product_T GROUP BY ProductFinish;
```

2. Use aggregate functions in WHERE clause

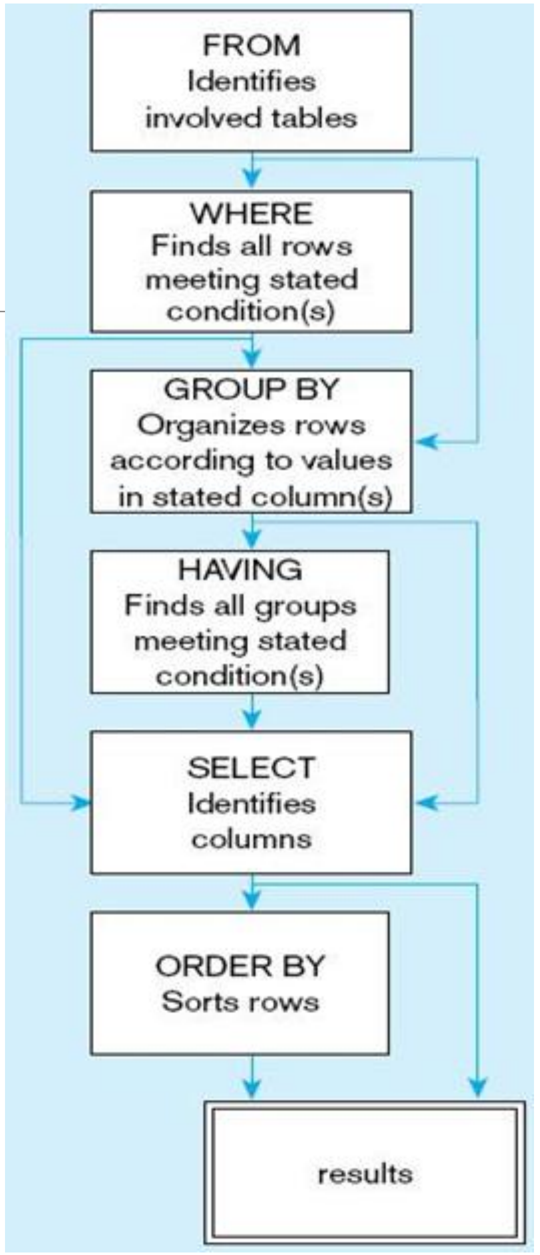
```
SELECT * FROM Product_T WHERE Price = MAX(Price);
```

3. SELECT attribute and aggregate function at the same time without a group by

```
SELECT ProductID, Max(ProductStandardPrice) FROM Product_T;
```

Steps to Design SQL Query

1. What tables are needed? [FROM clause]
2. How to connect these tables [JOIN or SUBQUERY]
3. What conditions should be met [WHERE clause]
4. Do I need to group the rows into categories? [GROUP BY clause]
5. Do I need to filter some groups? [HAVING clause]
6. Which columns do I want to display in the result? [SELECT clause]
7. Do I need to sort the results? [ORDER BY clause]
8. Write your SQL command following the syntax order


FROM
Identifies involved tables
WHERE
Finds all rows meeting stated condition(s)
GROUP BY
Organizes rows according to values in stated column(s)
HAVING
Finds all groups meeting stated condition(s)
SELECT
Identifies columns
ORDER BY
Sorts rows
results

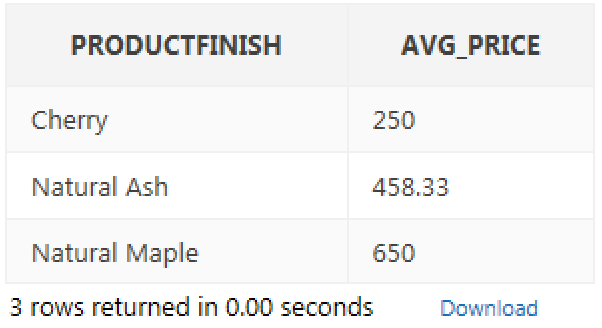
All-Together Example

Query: List in alphabetical order, the product finish and the average standard price for each finish that have an average standard price less than $750. We are only interested in the following finishes: Cherry, Natural Ash, Natural Maple, and White Ash

```
SELECT ProductFinish, ROUND(AVG(ProductStandardPrice), 2) AS Avg_Price
FROM Product_T
WHERE ProductFinish IN ('Cherry','Natural Ash','Natural Maple','White Ash')
GROUP BY ProductFinish
HAVING AVG(ProductStandardPrice) < 750
ORDER BY ProductFinish;
```

Note: You have to follow the order of the syntax when writing your SQL queries. Do not mess it up.

Output Table:



PRODUCTFINISH	AVG_PRICE
Cherry	250
Natural Ash	458.33
Natural Maple	650

3 rows returned in 0.00 seconds