# Oracle 18c and 19c

Relational Database Management System, widely used in enterprise applications.

# Introduction

- Oracle database is a relational database management system.
- It is also called OracleDB, or simply Oracle. It is produced and marketed by Oracle Corporation.
- It was created in 1977 by **Lawrence Ellison** and **Bob Miner**, **Ed Oates**, and **Bruce Scott** in August 1977.
- Database engines in the IT market for storing, organizing, and retrieving data.

# Introduction

- Oracle database was the first DB that designed for enterprise grid computing and data warehousing.
- Enterprise grid computing provides the most flexible and cost-effective way to manage information and applications.
- It uses SQL queries as a language for interacting with the database.

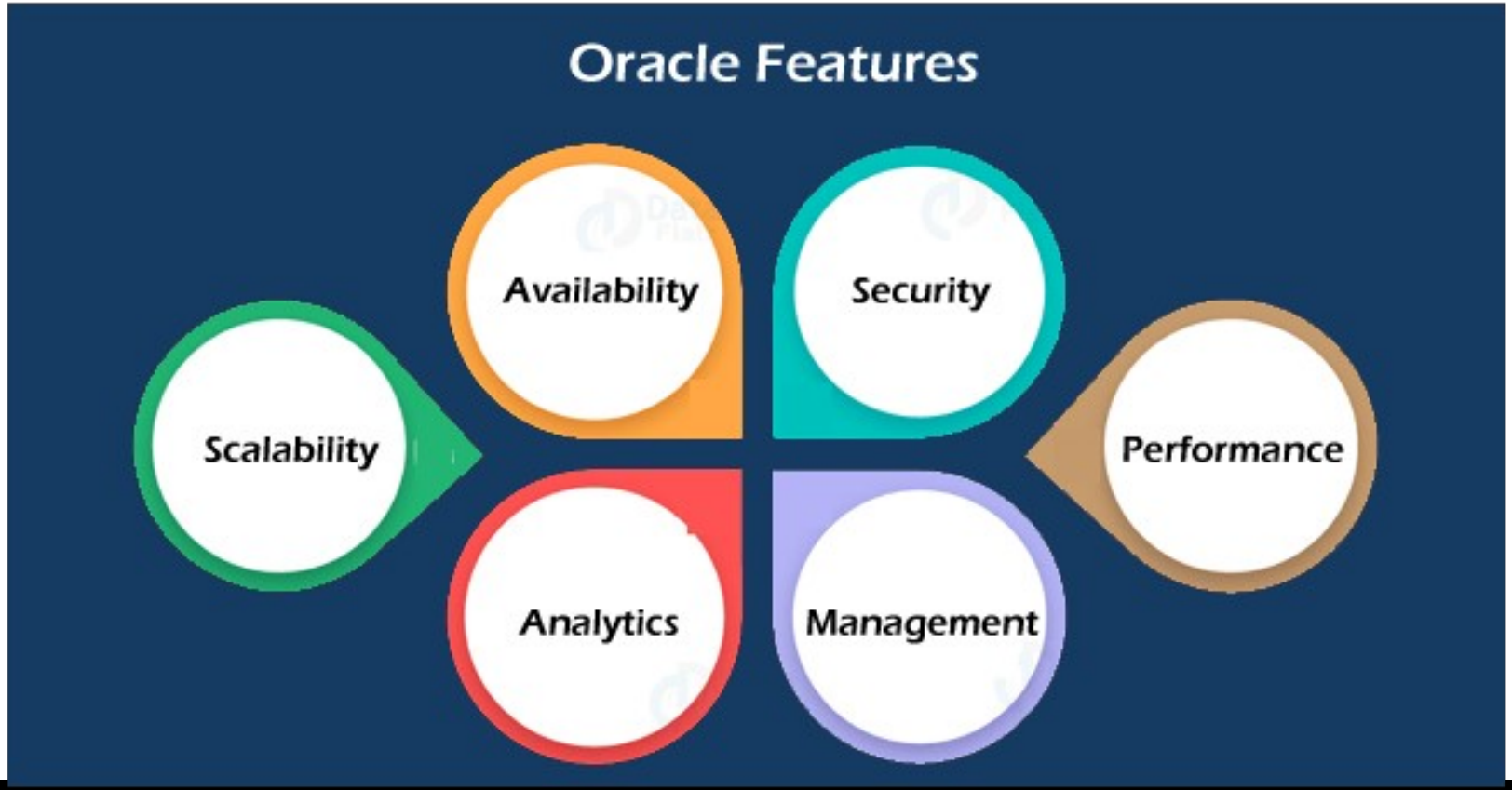
# Editions of Oracle database

- **Enterprise Edition:** It is the most robust and secure edition. It offers all features, including superior performance and security.
- **Standard Edition:** It provides the base functionality for users that do not require Enterprise Edition's robust package.
- **Express Edition (XE):** It is the lightweight, free and limited Windows, and Linux edition.
- **Oracle Lite:** It is designed for mobile devices.
- **Personal Edition:** It's comparable to the Enterprise Edition but without the Oracle Real Application Clusters feature.

# What is Oracle?

- Oracle was named after **"Project Oracle"** a project for one of their clients named **Central Intelligence Agency**, and the company that created Oracle was called **Systems Development Labs** (SDL).
- Systems Development Labs was renamed **Relational Software Inc**. (RSI) in 1978 to expand their market for the new database.
- They had again changed the name of the company from RSI to **Oracle Systems Corporation** in **1982**.

# Features of Oracle



# Benefits is Oracle?

- **Performance:** Oracle has procedures and principles that help us to get high levels of database performance. We can increase query execution time and operations with the use of performance optimization techniques in its database. This technique helps to retrieve and alter data faster.
- **Portability:** The Oracle database can be ported on all different platforms than any of its competitors. We can use this database on around 20 networking protocols as well as over 100 hardware platforms. This database makes it simple to write an Oracle application by making changes to the OS and hardware in a secure manner.
- **Backup and Recovery:** It is always better to take a proper backup of your entire oracle online backup and recovery. The Oracle database makes it easy to accomplish recovery quickly by using the. RMAN (Recovery Manager) functionality. It can recover or restore database files during downtime or outages. It can be used for online backups, archived backups, and continuous archiving. We can also use SQL* PLUS for recovery, which is known as user-managed recovery.

# Benefits is Oracle?

- **PL/SQL:** One of the greatest benefits of using the Oracle database is to support PL/SQL extension for procedural programming.
- **Multiple Database:** Oracle database allows several database instances management on a single server. It provides an instance caging approach for managing CPU allocations on a server hosting database instances. The database resource management and instance caging can work together to manage services across multiple instances.
- **Flashback Technology:** This advantage comes with the recent Oracle version. It allows us to recover those data that are incorrectly deleted or lost by human errors like accidental deletion of valuable data, deleting the wrong data, or dropping the table.

# Disadvantages of Oracle Database

- **Complexity:** Oracle is not recommended to use when the users are not technically savvy and have limited technical skills required to deal with the Oracle Database. It is also not advised to use if the company is looking for a database with limited functionality and easy to use.
- **Cost:** The price of Oracle products is very high in comparison to other databases. Therefore users are more likely to choose other less expensive options such as MS SQL Server, MySQL, etc.
- **Difficult to manage:** Oracle databases are often much more complex and difficult in terms of the management of certain activities.

# Instllatation

- timedatectl
- lsb_release -a
- sudo apt-get install -y vim net-tools openssh-server
- hostnamectl

# Required Packages

- sudo apt install alien autoconf automake autotools-dev binutils
- sudo apt install bzip2 doxygen elfutils expat gawk gcc gcc-multilib g++-multilib
- sudo apt install libelf-dev libltdl-dev libodbcinstq4-1 libodbcinstq4-1:i386
- sudo apt install libpth-dev libpthread-stubs0-dev libstdc++5  make
- sudo apt install rlwrap rpm sysstat unixodbc unixodbc-dev unzip
- sudo apt install x11-utils zlibc libaio1 libaio-dev ia32-libs openssh-server

# CREATE TABLE

To create a table, you have to name that table and define its columns and datatype for each column.

```
CREATE TABLE table_name
(
  column1 datatype [ NULL | NOT NULL ],
  column2 datatype [ NULL | NOT NULL ],
  ...
  column_n datatype [ NULL | NOT NULL ]
);
```

```
CREATE TABLE customers
( customer_id number(10) NOT NULL,
  customer_name varchar2(50) NOT NULL,
  city varchar2(50)
);
```

# Primary key

A primary key is a single field or combination of fields that contains a unique record. It must be filled. None of the field of primary key can contain a null value. A table can have only one primary key.

```
CREATE TABLE customers
( customer_id number(10) NOT NULL,
  customer_name varchar2(50) NOT NULL,
  city varchar2(50),
  CONSTRAINT customers_pk PRIMARY KEY (customer_id)
);
```

# CREATE TABLE AS Statement

The CREATE TABLE AS statement is used to create a table from an existing table by copying the columns of existing table.

CREATE TABLE new_table AS (SELECT * FROM old_table);

CREATE TABLE newcustomers AS (SELECT * FROM customers  WHERE customer_id < 5000);

# copying selected columns of another table

```
CREATE TABLE new_table
    AS (SELECT column_1, column2, ... column_n
        FROM old_table);
```

```
CREATE TABLE newcustomers2
AS (SELECT customer_id, customer_name
    FROM customers
    WHERE customer_id < 5000);
```

# copying selected columns from multiple tables

CREATE TABLE new_table AS (SELECT column_1, column2, ... column_n FROM old_table_1, old_table_2, ... old_table_n);

CREATE TABLE "regularcustomers" ( "RCUSTOMER_ID" NUMBER(10,0) NOT NULL ENABLE, "RCUSTOMER_NAME" VARCHAR2(50) NOT NULL ENABLE, "RC_CITY" VARCHAR2(50) );

CREATE TABLE "irregularcustomers" ( "IRCUSTOMER_ID" NUMBER(10,0) NOT NULL ENABLE, "IRCUSTOMER_NAME" VARCHAR2(50) NOT NULL ENABLE, "IRC_CITY" VARCHAR2(50));

# copying selected columns from multiple tables

```
CREATE TABLE newcustomers3
AS (SELECT regularcustomers.rcustomer_id,
regularcustomers.rc_city, irregularcustomers.ircustomer_name

FROM regularcustomers, irregularcustomers
WHERE regularcustomers.rcustomer_id =
irregularcustomers.ircustomer_id
AND regularcustomers.rcustomer_id < 5000);
```

# Oracle ALTER TABLE Statement

ALTER TABLE statement specifies how to add, modify, drop or delete columns in a table. It is also used to rename a table

*ALTER TABLE table_name*

*ADD column_name column-definition;*

ALTER TABLE customers

ADD customer_age varchar2(50);

# Oracle ALTER TABLE Statement

```
ALTER TABLE table_name
  ADD (column_1 column-definition,
       column_2 column-definition,
       ...
       column_n column_definition);
```

```
ALTER TABLE customers
  ADD (customer_type varchar2(50),
       customer_address varchar2(50));
```

# Oracle MODIFY TABLE Statement

```
ALTER TABLE table_name  MODIFY column_name column_type;
```

```
ALTER TABLE customers  MODIFY customer_name varchar2(100) not null;
```

```
ALTER TABLE table_name  MODIFY (column_1 column_type,
        column_2 column_type,
        ...
        column_n column_type);
```

```
ALTER TABLE customers  MODIFY (customer_name varchar2(100) not null,
        city varchar2(100));
```

# Oracle DROP TABLE Statement

```
ALTER TABLE table_name
  DROP COLUMN column_name;
```

```
ALTER TABLE customers
    DROP COLUMN customer_name;
```

# Oracle RENAME COLUMN

ALTER TABLE table_name RENAME COLUMN old_name to new_name;

ALTER TABLE customers RENAME COLUMN customer_name to cname;

ALTER TABLE table_name  RENAME TO new_table_name;

ALTER TABLE customers RENAME TO retailers;

# Oracle DROP TABLE Statement

```
DROP [schema_name].TABLE table_name [ CASCADE CONSTRAINTS ]
[ PURGE ];
```

**schema_name:** It specifies the name of the schema that owns the table.

**table_name:** It specifies the name of the table which you want to remove from the Oracle database.

**CASCADE CONSTRAINTS:** It is optional. If specified, it will drop all referential integrity constraints as well.

**PURGE:** It is also optional. If specified, the table and its dependent objects are placed in the recycle bin and can?t be recovered.

```
DROP TABLE customers;
```

```
DROP TABLE customers PURGE;
```

# Oracle Global Temporary tables

- Temporary tables generally contain all of the features that ordinary tables have like triggers, join cardinality, information about rows and block etc. the main difference is that the temporary tables can't have foreign keys related to other tables.

```
CREATE GLOBAL TEMPORARY TABLE table_name
( column1 datatype [ NULL | NOT NULL ],
  column2 datatype [ NULL | NOT NULL ],
  ...
  column_n datatype [ NULL | NOT NULL ]
);
```

# Oracle Global Temporary tables

- The parameter **table_name** specifies the global temporary table that you want to create.
- **column1, column2, ... column_ n:** It specifies the column that you want create in the global temporary table.
- Every column must have a datatype and should be defined as **NULL** or **NOT NULL**. If the value is left blank, it is by default treated as **NULL**.

# Oracle Global Temporary tables

```
CREATE GLOBAL TEMPORARY TABLE students
( student_id numeric(10) NOT NULL,
  student_name varchar2(50) NOT NULL,
  student_address varchar2(50)
);
```

# Oracle Global Temporary tables

```
DECLARE LOCAL TEMPORARY TABLE table_name
( column1 datatype [ NULL | NOT NULL ],
  column2 datatype [ NULL | NOT NULL ],
  ...
  column_n datatype [ NULL | NOT NULL ]
);
```

**table_name:** The parameter table_name specifies the local temporary table that you want to create.

**column1, column2,... column_ n:** It specifies the column that you want create in the local temporary table. Every column must have a datatype and should be defined as NULL or NOTNULL. If the value is left blank, it is by default treated as NULL.

# Oracle View

The view is a virtual table that does not physically exist. It is stored in Oracle data dictionary and do not store any data. It can be executed when called.

```
CREATE VIEW view_name AS SELECT columns FROM tables WHERE conditions;
```

```
CREATE TABLE  "SUPPLIERS" ( "SUPPLIER_ID" NUMBER, "SUPPLIER_NAME" VARCHAR2(4000), "SUPPLIER_ADDRESS" VARCHAR2(4000));
```

```
CREATE TABLE  "ORDERS" ( "ORDER_NO." NUMBER, "QUANTITY" NUMBER, "PRICE" NUMBER);
```

```
CREATE VIEW sup_orders AS SELECT suppliers.supplier_id, orders.quantity, orders.price FROM suppliers INNER JOIN orders ON suppliers.supplier_id = supplier_id WHERE suppliers.supplier_name = 'VOJO';
```

```
SELECT * FROM sup_orders;
```

# Oracle Update VIEW

CREATE OR REPLACE VIEW statement is used to modify the definition of an Oracle VIEW without dropping it.

```
CREATE OR REPLACE VIEW view_name AS SELECT columns FROM table WHERE conditions;
```

Execute the following query to update the definition of Oracle VIEW called sup_orders without dropping it.

```
CREATE or REPLACE VIEW sup_orders AS  SELECT suppliers.supplier_id, orders.quantity, orders.price FROM suppliers INNER JOIN orders ON suppliers.supplier_id = supplier_id WHERE suppliers.supplier_name = 'HCL';
```

# Oracle Drop VIEW

```
SELECT * FROM sup_orders;

DROP VIEW view_name;

DROP VIEW sup_orders;
```

# Oracle Queries

- SELECT * from customers;
- insert into customers values(101,'rahul','delhi');
- update customers set name='bob', city='london' where id=101;
- delete from customers where id=101;
- truncate table customers;
- drop table customers;
- CREATE TABLE customers ( id number(10) NOT NULL, name varchar2(50) NOT NULL, city varchar2(50), CONSTRAINT customers_pk PRIMARY KEY (id) );

# Oracle Queries

- ALTER TABLE customers ADD age varchar2(50);
- SELECT expressions FROM tables WHERE conditions;
- SELECT * FROM customers;
- SELECT age, address, salary FROM customers WHERE  age < 25 AND salary > '20000' ORDER BY age ASC, salary DESC;
- SELECT customers.name, courses.trainer FROM courses INNER JOIN customers ON courses.course_id = course_id ORDER BY name;

# Oracle Insert Statement

- INSERT INTO table (column1, column2, ... column_n ) VALUES (expression1, expression2, ... expression_n );
- INSERT INTO table (column1, column2, ... column_n )
- SELECT expression1, expression2, ... expression_n FROM source_table WHERE conditions;
- INSERT INTO suppliers (supplier_id, supplier_name) VALUES (50, 'Flipkart');
- INSERT INTO suppliers (supplier_id, supplier_name) SELECT age, address FROM customers WHERE age > 20;
- SELECT count(*) FROM customers WHERE age > 20;

# Oracle INSERT ALL statement

- INSERT ALL
  INTO table_name (column1, column2, column_n) VALUES (expr1, expr2, expr_n)
  INTO table_name(column1, column2, column_n) VALUES (expr1, expr2, expr_n)
  INTO table_name (column1, column2, column_n) VALUES (expr1, expr2, expr_n)
- SELECT * FROM dual;
- INSERT ALL
  INTO suppliers (supplier_id, supplier_name) VALUES (20, 'Google')
  INTO suppliers (supplier_id, supplier_name) VALUES (21, 'Microsoft')
  INTO suppliers (supplier_id, supplier_name) VALUES (22, 'Apple')
- SELECT * FROM dual;

# Oracle INSERT ALL statement

- INSERT INTO suppliers (supplier_id, supplier_name) VALUES (1000, 'Google');
- INSERT INTO suppliers (supplier_id, supplier_name) VALUES (2000, 'Microsoft');
- INSERT INTO suppliers (supplier_id, supplier_name) VALUES (3000, 'Apple');
- INSERT ALL
  INTO suppliers (supplier_id, supplier_name) VALUES (30, 'Google')
  INTO suppliers (supplier_id, supplier_name) VALUES (31, 'Microsoft')
  INTO customers (age, name, address) VALUES (29, 'Luca Warsi', 'New York')
- SELECT * FROM dual;

# Oracle UPDATE Statement

- UPDATE table SET column1 = expression1, column2 = expression2,

  ...

  column_n = expression_n  WHERE conditions;
- UPDATE table1 SET column1 = (SELECT expression1 FROM table2 WHERE conditions) WHERE conditions;

# Oracle Update Example

- UPDATE suppliers SET supplier_name = 'Kingfisher' WHERE supplier_id = 2;
- UPDATE suppliers SET supplier_address = 'Agra', supplier_name = 'Bata shoes' WHERE supplier_id = 1;
- UPDATE customers SET name = (SELECT supplier_name FROM suppliers WHERE suppliers.supplier_name = customers.name) WHERE age < 25;  v

# Oracle DELETE Statement

- DELETE FROM table_name WHERE conditions;
- DELETE FROM customers WHERE name = 'Sohan';
- DELETE FROM customers WHERE last_name = 'Maurya' AND customer_id > 2;
- TRUNCATE TABLE [schema_name.]table_name
- TRUNCATE TABLE customers;
- DELETE TABLE customers;

# Oracle DISTINCT Clause

- SELECT DISTINCT expressions FROM tables WHERE conditions;
- CREATE TABLE  "CUSTOMERS" ( "NAME" VARCHAR2(4000), "AGE" NUMBER, "SALARY" NUMBER, "STATE" VARCHAR2(4000));
- SELECT DISTINCT state FROM customers WHERE name = 'charu';
- SELECT DISTINCT name, age, salary FROM customers WHERE age >= '60';

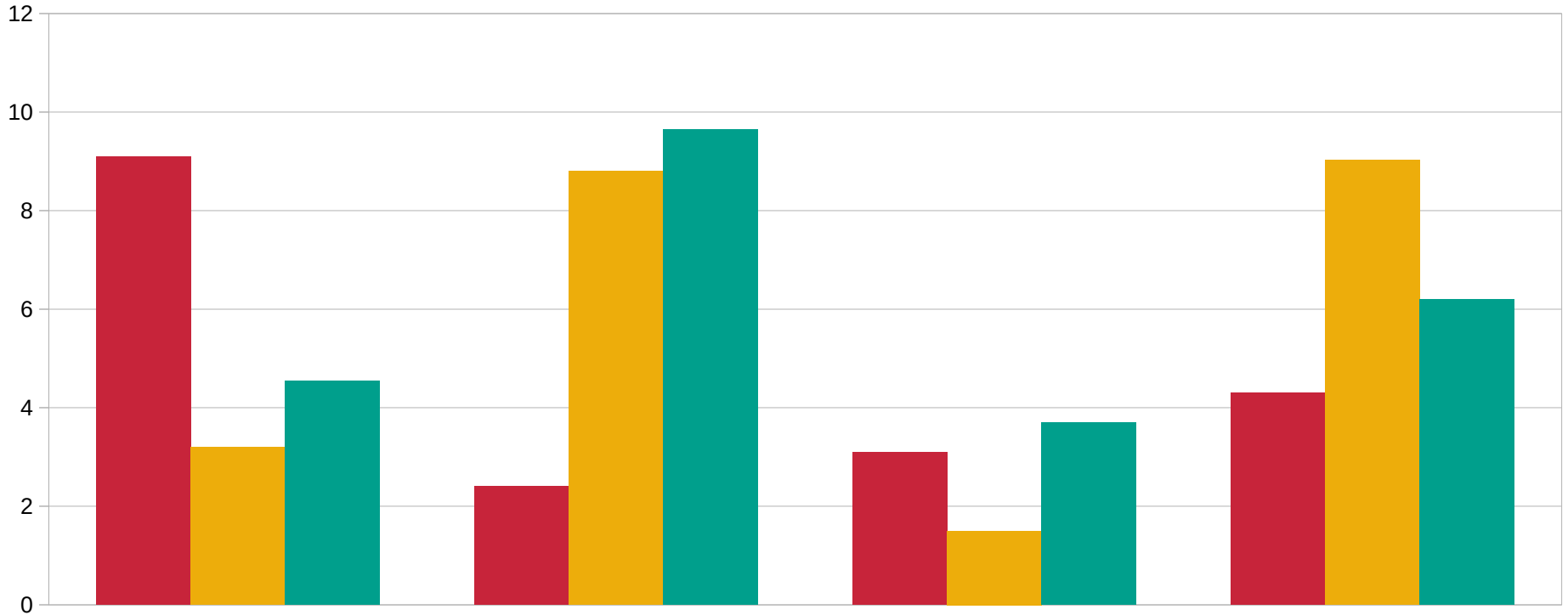
# Oracle ORDER BY Clause

- SELECT expressions FROM tables WHERE conditions ORDER BY expression [ ASC | DESC ];
- CREATE TABLE  "SUPPLIER" ( "SUPPLIER_ID" NUMBER, "FIRST_NAME" VARCHAR2(4000), "LAST_NAME" VARCHAR2(4000) );
- SELECT * FROM supplier ORDER BY last_name;
- SELECT * FROM supplier ORDER BY last_name DESC;

# Oracle GROUP BY Clause

- SELECT expression1, expression2, ... expression_n, aggregate_function (aggregate_expression) FROM tables WHERE conditions GROUP BY expression1, expression2, ... expression_n;
- CREATE TABLE  "SALESDEPARTMENT" ( "ITEM" VARCHAR2(4000), "SALE" NUMBER, "BILLING_ADDRESS" VARCHAR2(4000) );
- SELECT item, SUM(sale) AS "Total sales" FROM salesdepartment GROUP BY item;
- CREATE TABLE  "CUSTOMERS" ( "NAME" VARCHAR2(4000), "AGE" NUMBER, "SALARY" NUMBER, "STATE" VARCHAR2(4000) );
-
- SELECT state, COUNT(*) AS "Number of customers"
- FROM customers
- WHERE salary > 10000
- GROUP BY state;

| | | | | |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |


12
10
8
6
4
2
0