Department of Software Engineering
Mehran University of Engineering and Technology, Jamshoro

Course: SW215 – Database System			
Instructor	Ms Shafiya Qadeer	**Practical/Lab No.**	09
Date	24-02-2021	**CLOs**	2
Signature		**Assessment Score**	2 Marks

Topic	To become familiar with views and indexes.
.Objectives	- To become familiar with Simple & Complex Views, Indexex

Lab Discussion: Theoretical concepts and Procedural steps

VIEWS

Views subsets of data from one or more tables

- VIEW is a virtual table that does not physically exist.
- A view is based on the result-set of an SQL statement.
- A view contains no data of its own but is like a window through which data from tables can be viewed or changed.
- The tables on which a view is based are called base tables.
- A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.
- It is created by a query joining one or more tables.
- WHY USE VIEWS ?
  - To restrict data access (can display only selective columns to user)
  - To make complex queries easy

TYPES OF VIEWS

- 2 types of views:

1. Simple Views
2. Complex Views

SIMPLE VIEW

- A simple view is one that:
    - ➢ Derives data from only one table.
    - ➢ Contains no functions or groups of data.
    - ➢ Can perform DML operations through the view.

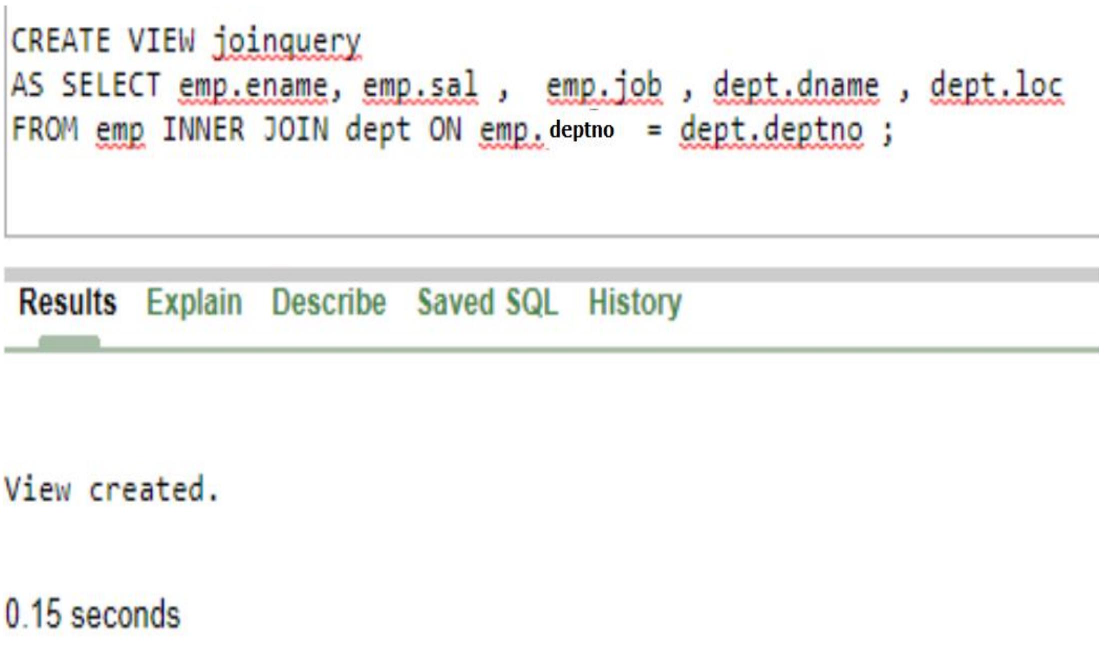
COMPLEX VIEW

- A complex view is one that:
    - ➢ Derives data from many tables.
    - ➢ Contains functions or groups of data.

Does not always allow DML operations through the view

SYNTAX:

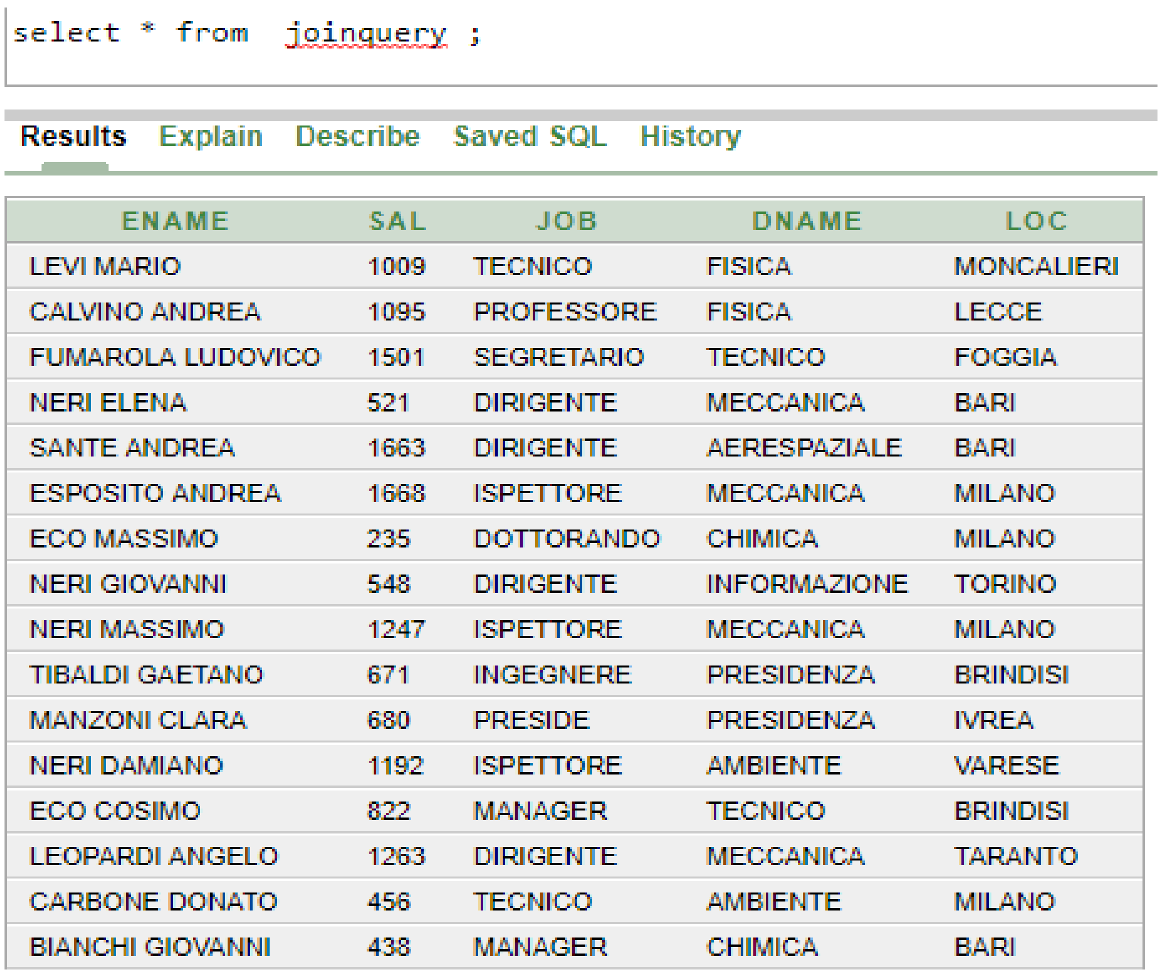
```
CREATE VIEW view_name AS
        SELECT columns
        FROM tables
        WHERE conditions ;
```



```
CREATE VIEW joinquery
AS SELECT emp.ename, emp.sal ,  emp.job , dept.dname , dept.loc
FROM emp INNER JOIN dept ON emp.deptno  = dept.deptno ;
```

```
View created.

0.15 seconds
```

RETRIEVING DATA FROM A VIEW


```
select * from  joinquery ;
```

ENAME	SAL	JOB	DNAME	LOC
LEVI MARIO	1009	TECNICO	FISICA	MONCALIERI
CALVINO ANDREA	1095	PROFESSORE	FISICA	LECCE
FUMAROLA LUDOVICO	1501	SEGRETARIO	TECNICO	FOGGIA
NERI ELENA	521	DIRIGENTE	MECCANICA	BARI
SANTE ANDREA	1663	DIRIGENTE	AERESPAZIALE	BARI
ESPOSITO ANDREA	1668	ISPETTORE	MECCANICA	MILANO
ECO MASSIMO	235	DOTTORANDO	CHIMICA	MILANO
NERI GIOVANNI	548	DIRIGENTE	INFORMAZIONE	TORINO
NERI MASSIMO	1247	ISPETTORE	MECCANICA	MILANO
TIBALDI GAETANO	671	INGEGNERE	PRESIDENZA	BRINDISI
MANZONI CLARA	680	PRESIDE	PRESIDENZA	IVREA
NERI DAMIANO	1192	ISPETTORE	AMBIENTE	VARESE
ECO COSIMO	822	MANAGER	TECNICO	BRINDISI
LEOPARDI ANGELO	1263	DIRIGENTE	MECCANICA	TARANTO
CARBONE DONATO	456	TECNICO	AMBIENTE	MILANO
BIANCHI GIOVANNI	438	MANAGER	CHIMICA	BARI

CHECK WHETHER VIEW IS CREATED OR NOT

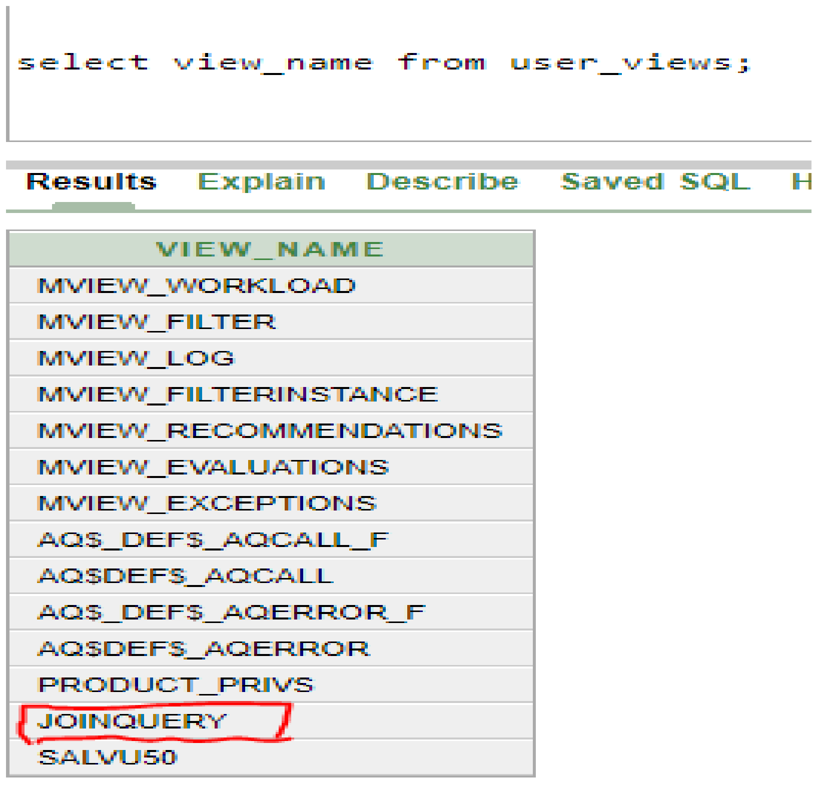
1. user_views:

Used to display all the views owned by logged user.

2. all_views:

Used to display all the views accessible by logged user.

    select view_name from   all_views ;



```
select view_name from user_views;
```

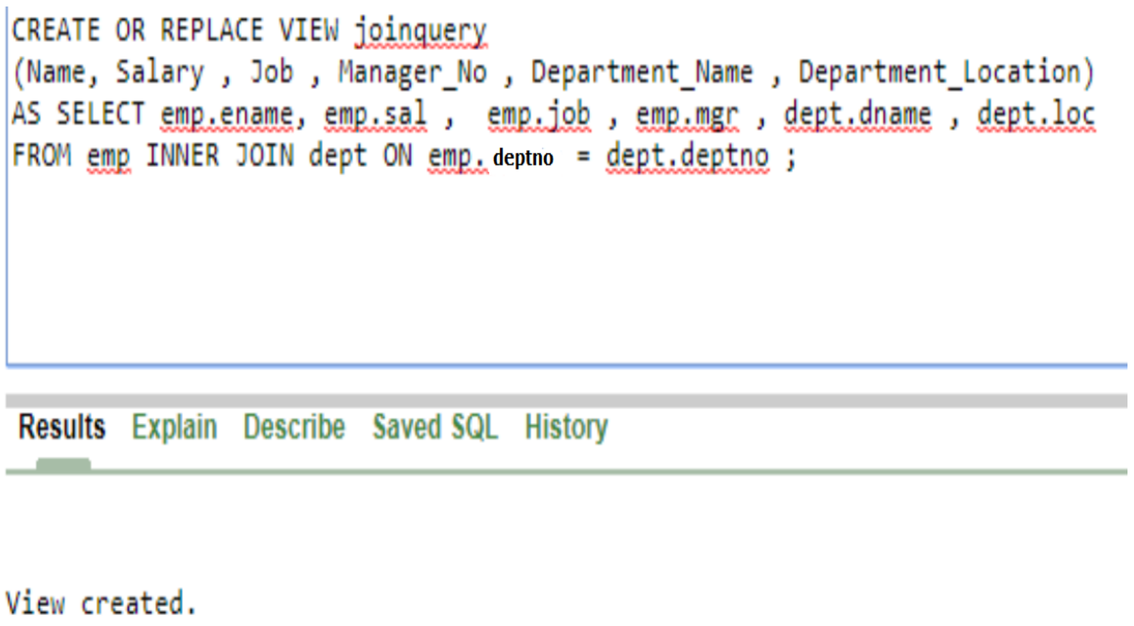
VIEW_NAME
MVIEW_WORKLOAD
MVIEW_FILTER
MVIEW_LOG
MVIEW_FILTERINSTANCE
MVIEW_RECOMMENDATIONS
MVIEW_EVALUATIONS
MVIEW_EXCEPTIONS
AQ$_DEF$_AQCALL_F
AQ$DEF$_AQCALL
AQ$_DEF$_AQERROR_F
AQ$DEF$_AQERROR
PRODUCT_PRIVS
JOINQUERY
SALVU50

REPLACING/ALTERING VIEWS:

- A view can be dropped and then re-created. When a view is dropped, all grants of corresponding view privileges are revoked from roles and users. After the view is re-created, necessary privileges must be re-granted.
- **SYNTAX:**DROP VIEW view_name ;

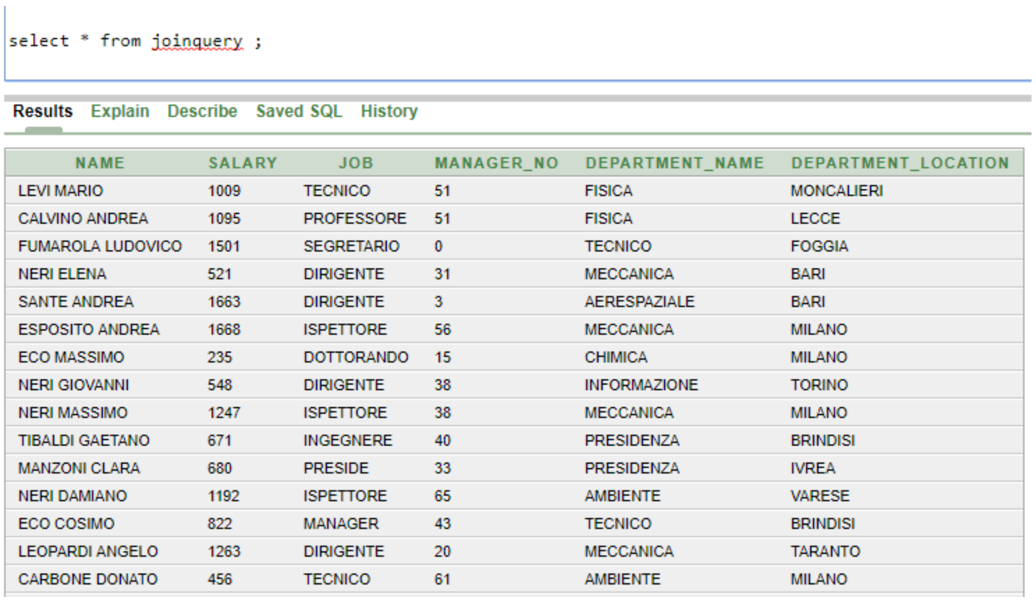
OR REPLACE OPTION WITH THE CREATE VIEW CLAUSE

- With the OR REPLACE option, a view can be created even if one exists with the same name already, thus replacing the old version of the view for its owner.
- No need to drop , recreate and regrant obect privileges to a view.



```
CREATE OR REPLACE VIEW joinquery
(Name, Salary , Job , Manager_No , Department_Name , Department_Location)
AS SELECT emp.ename, emp.sal ,  emp.job , emp.mgr , dept.dname , dept.loc
FROM emp INNER JOIN dept ON emp.deptno  = dept.deptno ;
```

```
View created.
```

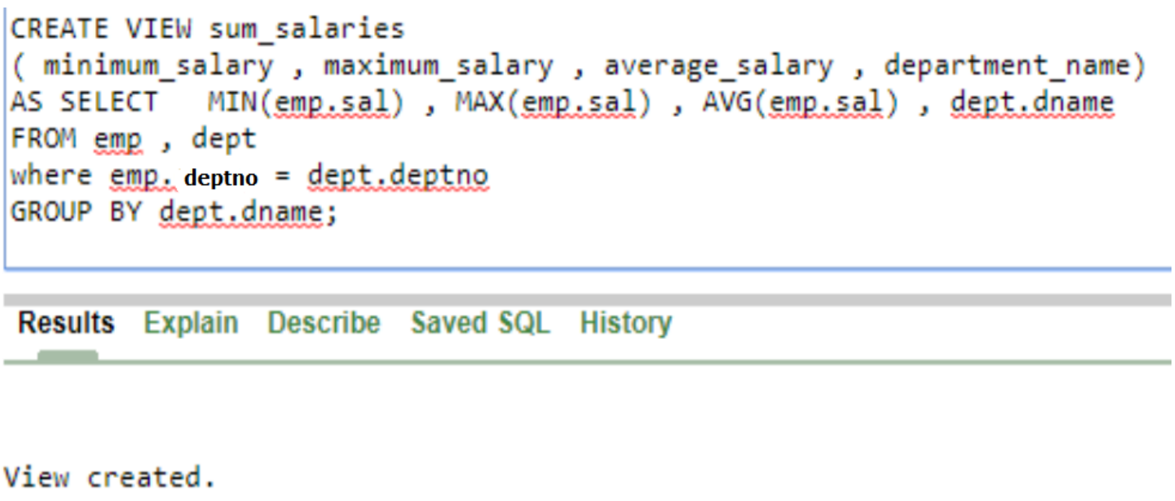


```
select * from joinquery ;
```

NAME	SALARY	JOB	MANAGER_NO	DEPARTMENT_NAME	DEPARTMENT_LOCATION
LEVI MARIO	1009	TECNICO	51	FISICA	MONCALIERI
CALVINO ANDREA	1095	PROFESSORE	51	FISICA	LECCE
FUMAROLA LUDOVICO	1501	SEGRETARIO	0	TECNICO	FOGGIA
NERI ELENA	521	DIRIGENTE	31	MECCANICA	BARI
SANTE ANDREA	1663	DIRIGENTE	3	AERESPAZIALE	BARI
ESPOSITO ANDREA	1668	ISPETTORE	56	MECCANICA	MILANO
ECO MASSIMO	235	DOTTORANDO	15	CHIMICA	MILANO
NERI GIOVANNI	548	DIRIGENTE	38	INFORMAZIONE	TORINO
NERI MASSIMO	1247	ISPETTORE	38	MECCANICA	MILANO
TIBALDI GAETANO	671	INGEGNERE	40	PRESIDENZA	BRINDISI
MANZONI CLARA	680	PRESIDE	33	PRESIDENZA	IVREA
NERI DAMIANO	1192	ISPETTORE	65	AMBIENTE	VARESE
ECO COSIMO	822	MANAGER	43	TECNICO	BRINDISI
LEOPARDI ANGELO	1263	DIRIGENTE	20	MECCANICA	TARANTO
CARBONE DONATO	456	TECNICO	61	AMBIENTE	MILANO

Creating A Complex View

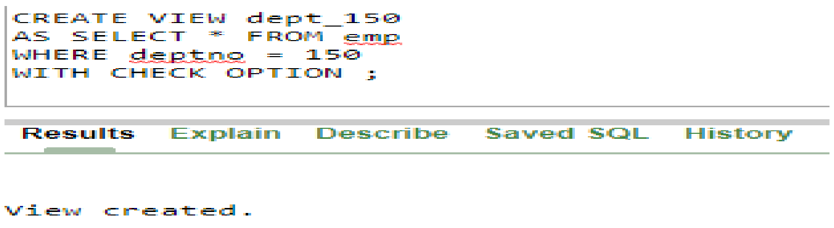
- It contains group functions to display values from 2 tables.
- Display dept names, min sal , max sal, avg sal by each department.



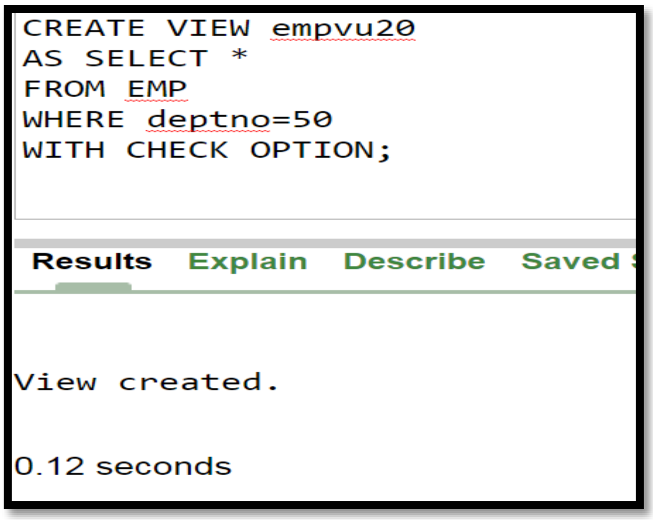
```
CREATE VIEW sum_salaries
( minimum_salary , maximum_salary , average_salary , department_name)
AS SELECT   MIN(emp.sal) , MAX(emp.sal) , AVG(emp.sal) , dept.dname
FROM emp , dept
where emp. deptno = dept.deptno
GROUP BY dept.dname;
```

```
View created.
```

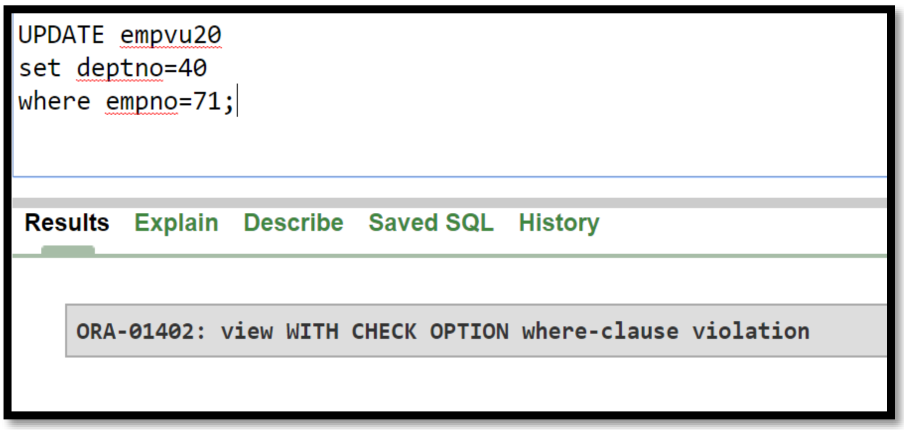
- The WITH CHECK OPTION is a CREATE VIEW statement option. The purpose of the WITH CHECK OPTION is to ensure that all UPDATE and INSERTs satisfy the condition(s) in the view definition.
- If they do not satisfy the condition(s), the UPDATE or INSERT returns an error.


```
CREATE VIEW dept_150
AS SELECT * FROM emp
WHERE deptno = 150
WITH CHECK OPTION ;
```

```
View created.
```


```
CREATE VIEW empvu20
AS SELECT *
FROM EMP
WHERE deptno=50
WITH CHECK OPTION;
```

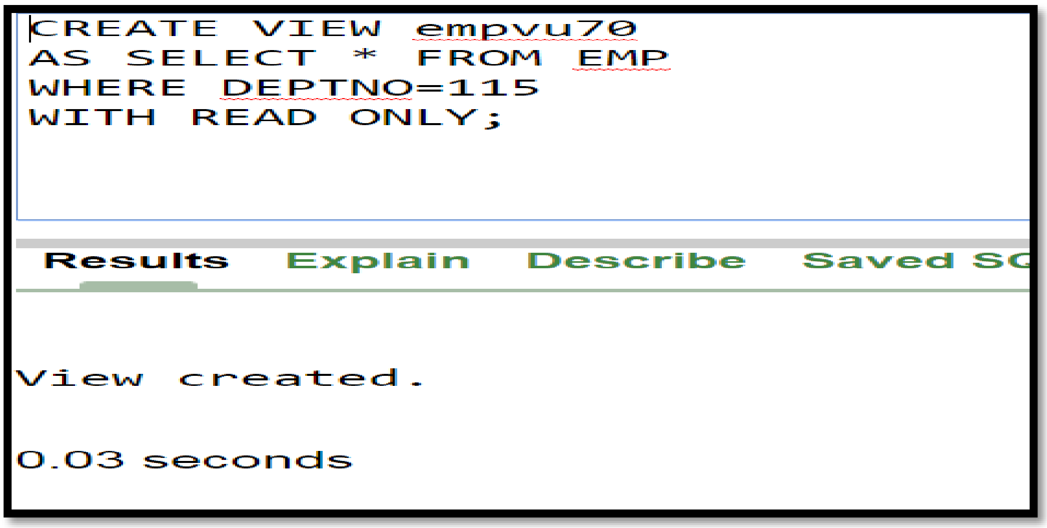
```
View created.

0.12 seconds
```


```
UPDATE empvu20
set deptno=40
where empno=71;
```

```
ORA-01402: view WITH CHECK OPTION where-clause violation
```

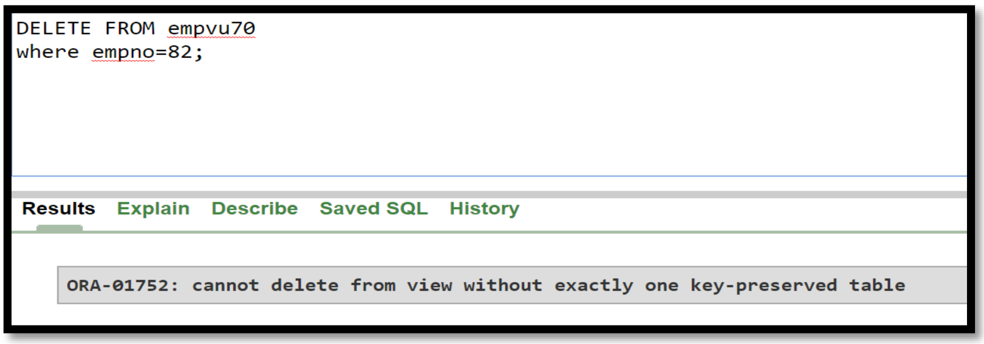
- You can ensure that no DML operations occur by adding the WITH READ ONLY option to your view definition.
- Any attempt to perform DML on any row in the view results in an Oracle server error.

<u>Denying DML operations:</u>



```
CREATE VIEW empvu70
AS SELECT * FROM EMP
WHERE DEPTNO=115
WITH READ ONLY;
```

```
View created.

0.03 seconds
```



```
DELETE FROM empvu70
where empno=82;
```

```
ORA-01752: cannot delete from view without exactly one key-preserved table
```

<u>INDEXES</u>

- An index is a pointer to data in a table.
- Is used by the oracle server to speed up the retrieval of rows by using a pointer.
- An index helps to speed up SELECT queries and WHERE clauses.

Indexes are physically and logically independent of the table they index. This means they can be created or dropped at any time and have no effect on the base tables or other indexes.

<u>TYPES</u>:

- Two types of indexes can be created .

1. **<u>unique index</u>**: the Oracle server automatically creates this index when you define a column in a table to have a PRIMARY KEY or UNIQUE key constraint. The name of the index is the name given to the constraint.
2. **<u>non unique index:</u>** which a user can create.

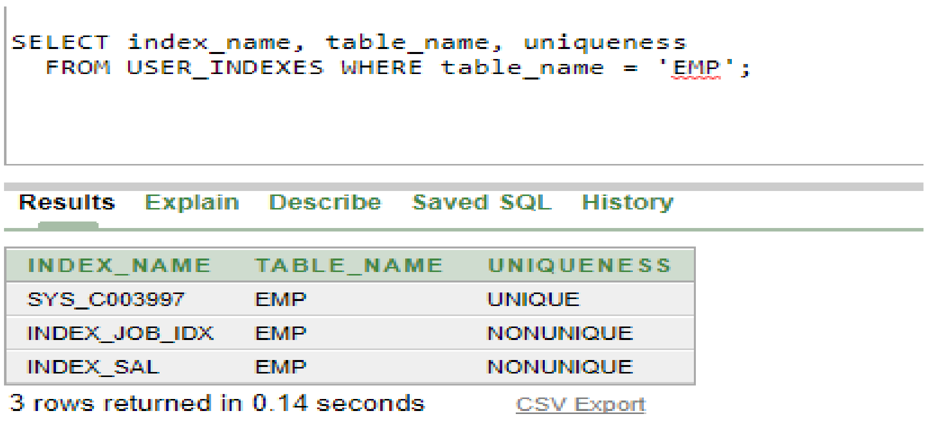
- Confirm the existence of indexes from the USER_INDEXES view.

- You can also check the columns involved in an index by querying the USER_IND_COLUMNS view.

Confirming Indexes

- Confirm the existence of indexes from the USER_INDEXES view.
- You can also check the columns involved in an index by querying the USER_IND_COLUMNS view.

USER_INDEXES describes the indexes owned by the current user. This view does not display the OWNER column



```
SELECT index_name, table_name, uniqueness
  FROM USER_INDEXES WHERE table_name = 'EMP';
```

INDEX_NAME	TABLE_NAME	UNIQUENESS
SYS_C003997	EMP	UNIQUE
INDEX_JOB_IDX	EMP	NONUNIQUE
INDEX_SAL	EMP	NONUNIQUE

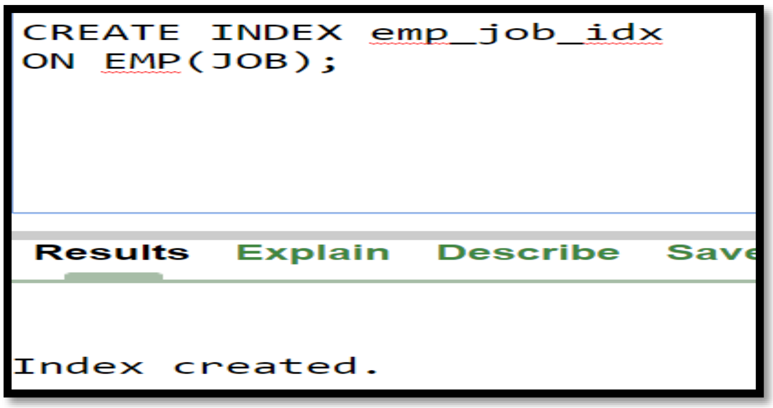
3 rows returned in 0.14 seconds

ALL_INDEXES describes the indexes on the tables accessible to the current user

SYNTAX:

CREATE INDEX index_name

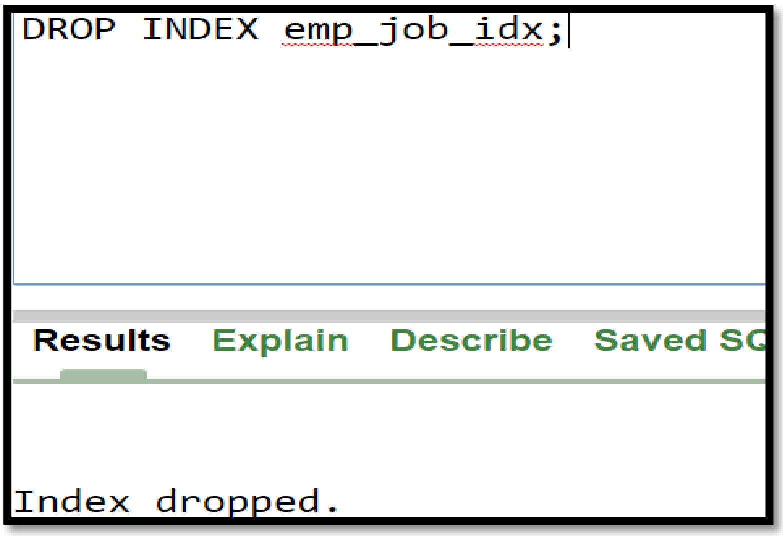
ON table_name (column1, column2, ... column_n);



```
CREATE INDEX emp_job_idx
ON EMP(JOB);
```

```
Index created.
```

Drop INDEX index_name ;



```
DROP INDEX emp_job_idx;
```

```
Index dropped.
```

<u>Index Guidelines:</u>

- indexes should not be used on small tables.
- Create primary keys for all tables, index will be created by default.
- Index the columns that are involved in multi-table join operations
- Index columns that are used frequently in where clauses.
- Index columns that are involved in order by, group by, union and distinct operations.
- Columns that are frequently update are bad for indexing
- Choose tables where few rows have similar values

Lab Tasks

1. Create a view call employee_vu based on the employee numbers, employee names and department numbers from the emp table. Change the heading for the ename to EMPLOYEE.
2. Display the contents of the employee_vu view.
3. Create a view called SALARY_VU based on the employee names, salaries and salary grades for all employees. Use the EMP and SALGRADE tables. Label the columns Emploee, salary and grade respectively.
4. Display the structure and contents of the DEPT50 view.
5. Test your view by reassigning "Levi Mario" (employee name) to department 80.
6. Create an index on the deptno of the Emp table. Drop the newly created index.
7. Create an index on the job and sal of the Emp table. Drop the newly created index.