**ORACLE SQL**

**DATABASE OBJECTS:**

* Two types: Schema objects or non-schema objects
* **SCHEMA OBJECTS:**
  + Schema is collection of logical structure of data and objects
  + There are many types of schema objects
  + **TABLES---** is base unit to store formatted data in form of rows and columns
  + **SYNONYM**—is a alternative name for database objects, are references to original objects
    - Also, we can query different objects without specifying the schema name
  + **VIEW**—is a virtual name, that provides access to a subset of columns or rows from one or more tables.
    - View is a virtual table that does not occupies any physical space
    - Is just an sql script when we run the query it executes and retrieves the returning data from query
    - Benefit is we can store the sql query and execute the view itself everywhere where we want to use.
  + **MATERIALIZED VIEW**— is a logic beyond the views
    - Is a real table that has the data filled in by the sql query
    - Use is that consider we have a long sql code tht takes long time to execute so everytime we execute the view it will take time, with help if this we can store that query’s result in a table and query the table every time if we need anything, this reduces the time
  + **CONSTRAINTS**—are the rules for restricting the entry of data in tables
  + **DATABASE LINKS**—is a connection between two database servers
    - We can link two database in same network
    - If we use more than one database in a server than we can use database links to reach to another database objects.
  + **INDEXES**—is used for improving the speed for retrieval of data
    - We can think indexes just like the indexes in books
    - They work automatically, we cannot run it.
  + **FUNCTIONS AND PROCEDURES—**Functions must return something but procedures donot return anything
  + **PACKAGES**—are schema object that is compiled and stored in database
    - Consists of procedures, functions, cursors etc to perform more than one operation using functions and procedures
  + **TRIGGERS**: are compiled program units stored in db and executed for specific event(create, update, delete, alter..)
    - They are mainly used to perform a operation before/after/on the main operation
  + **SEQUENCE**: are the database objects to create unique triggers
    - Are generally used for automatic generated primary key values
    - Can be used by multiple users
    - Cannot be rolled back, once the count is increased, whether you use or not.
* **NON SCHEMA OBJECTS**
  + Are the objects that are not stored in database
  + These are not in schema
  + Example- directories, roles, tablespaces, users

**Analysing user\_objects and all\_objects views**

select \* from user\_objects;

select \* from cat;

select \* from user\_catalog;

select \* from all\_objects;

select \* from dba\_objects;

**Searching in tables (user\_tables)**

select \* from user\_tables;

select \* from all\_tables;

select \* from dba\_tables;

select \* from user\_tables where table\_name = 'LOCATIONS';

select \* from tabs;

**Querying and searching columns of tables (USER\_TAB\_COLUMNS)**

select \* from user\_tab\_columns;

select \* from cols;

select column\_name,data\_type,data\_length,nullable,data\_default,num\_nulls,avg\_col\_len,num\_distinct from user\_tab\_columns

where table\_name = 'EMPLOYEES';

**Retrieving constraints of a table**

select \* from USER\_CONSTRAINTS;

select owner,table\_name,constraint\_name,constraint\_type,search\_condition,r\_constraint\_name,delete\_rule,status,index\_name

from user\_constraints;

--Finding columns bound to a constraint

select \* from user\_cons\_columns;

select \* from user\_cons\_columns where table\_name = 'EMPLOYEES';

**Finding related tables using table comments**

Select \* from user\_tab\_comments;

Select \* from user\_col\_comments;

select \* from user\_tab\_comments where upper(comments) like '%EMPLOYEES%';

select \* from user\_COL\_comments where upper(comments) like '%EMPLOYEES%';

**Creating Sequences**

create sequence employee\_seq

start with 200

increment by 3

maxvalue 9999

cache 50

nocycle

order;

drop sequence employee\_seq;

---Modifying sequences

ALter sequence employee\_seq

increment by 2

maxvalue 500

nocache

nocycle

noorder;

**Dropping a sequence**

drop sequence employee\_seq;

**Using sequence**

select employee\_seq.currval from dual;

select employee\_seq.nextval from dual;

insert into employees (employee\_id,first\_name,last\_name,email,hire\_date,salary,job\_id)

values (employee\_seq.nextval,'Alex','Earnst','ALEXIS',sysdate,'2500','IT\_PROG');

DELETE FROM EMPLOYEES WHERE EMPLOYEE\_ID = EMPLOYEE\_SEQ.CURRVAL;

DELETE FROM EMPLOYEES WHERE EMPLOYEE\_ID = 221;

**using sequence as default value**

create table temp (t1 number default employee\_seq.nextval not null, t2 varchar2(50));

insert into temp (t2) values ('alex');

select \* from temp;

drop table temp;

**analyzing user sequences**

desc user\_sequences;

select \* from user\_sequences;

select \* from user\_sequences where sequence\_name = 'EMPLOYEE\_SEQ';

**Creating synonyms**

create synonym sy for sys.user\_objects;

select \* from sy;

drop synonym sy;

**using synonyms**

select \* from sy where objeCt\_type = 'TABLE';

**Analyzing user\_synonyms**

desc user\_synonyms;

**CREATING INDEXES**

select \* from employees;

create unique index temp\_idx

on employees (employee\_id);

create unique index temp\_idx

on employees (phone\_number);

create bitmap index temp\_idx

on employees (first\_name,last\_name);

CREATE TABLE employee\_temp

(employee\_id NUMBER(6) PRIMARY KEY USING INDEX

(CREATE INDEX temp\_idx ON

employee\_temp(employee\_id)),

first\_name VARCHAR2(20),

last\_name VARCHAR2(25));

drop table employee\_temp;

**Function-based indexes**

Create index first\_name\_idx on employees (upper(first\_name));

drop index first\_name\_idx;

select \* from employees where UPPER(first\_name) = 'ALEX';

**Multiple Indexes with the same set of colums**

CREATE INDEX temp\_idx

ON employees(first\_name, last\_name);

ALTER INDEX temp\_idx INVISIBLE;

CREATE BITMAP INDEX temp\_idx

ON employees(first\_name, last\_name);

**Removing indexes**

drop index temp\_idx;

**Analyzing user\_indexes**

select \* from user\_indexes;

select \* from all\_col\_comments where table\_name = 'USER\_INDEXES';

select \* from user\_ind\_columns where table\_name = 'EMPLOYEES' ORDER BY INDEX\_NAME;

**Creating views**

CREATE VIEW empvw90 AS

SELECT \* FROM EMPLOYEES WHERE DEPARTMENT\_ID = 90;

SELECT \* FROM EMPVW40;

CREATE VIEW empvw20 AS

SELECT EMPLOYEE\_ID,FIRST\_NAME,LAST\_NAME FROM EMPLOYEES WHERE DEPARTMENT\_ID = 20;

CREATE VIEW empvw30 AS

SELECT EMPLOYEE\_ID EID,FIRST\_NAME NAME,LAST\_NAME SURNAME FROM EMPLOYEES WHERE DEPARTMENT\_ID = 30;

CREATE VIEW EMPVW40 (EID,NAME,SURNAME,EMAIL) AS

SELECT EMPLOYEE\_ID,FIRST\_NAME,LAST\_NAME,EMAIL FROM EMPLOYEES

WHERE DEPARTMENT\_ID = 40;

**Modifying Views**

CREATE OR REPLACE VIEW EMPVW30 (EID,NAME,EMAIL,JOB\_ID,PHONE) AS

SELECT EMPLOYEE\_ID,FIRST\_NAME||' '||LAST\_NAME,EMAIL,JOB\_ID,PHONE\_NUMBER FROM EMPLOYEES

WHERE DEPARTMENT\_ID = 30;

SELECT \* FROM EMPVW30;

DROP VIEW EMPVW30;

**Performing DML Operations with views**

create view empvw80 as

select employee\_id,first\_name,last\_name,email,hire\_date,job\_id,department\_id from employees

where department\_id = '80'

with check option;

insert into empvw80 values

(208,'Alex', 'Brown','ABROWN',sysdate,'IT\_PROG',80);

insert into empvw80( employee\_id,first\_name,last\_name,email,hire\_date,job\_id) values

(207,'Alex', 'Brown','ABROWN',sysdate,'IT\_PROG');

delete from empvw80 where employee\_id = 207;

select \* from empvw80 where employee\_id = 207;

delete from employees where employee\_id = 207;

drop view empvw80;

**Preventing dml operations on a view**

create view empvw80 as

select employee\_id,first\_name,last\_name,email,hire\_date,job\_id,department\_id from employees

where department\_id = '80'

with read only;

insert into empvw80 values

(208,'Alex', 'Brown','ABROWN',sysdate,'IT\_PROG',80);

**Materialized Views**

CREATE MATERIALIZED VIEW deparment\_max\_salaries\_mv

BUILD IMMEDIATE

REFRESH COMPLETE ON DEMAND

ENABLE QUERY REWRITE AS

SELECT DEPARTMENT\_ID,max(salary) FROM employees

GROUP BY DEPARTMENT\_ID

ORDER BY DEPARTMENT\_ID;

CREATE MATERIALIZED VIEW LOG ON employees;

DROP MATERIALIZED VIEW LOG ON employees;

drop materialized view deparment\_max\_salaries\_mv;

**Fast Refresh**

Create materialized view vw\_sales\_managers

build immediate

refresh fast on commit

as SELECT \* FROM employees

WHERE job\_id = 'SA\_MAN';

SELECT \* FROM vw\_sales\_managers;

insert into employees values (400,'Alex','Brown','abrown','111111111',sysdate,'SA\_MAN',10000,NULL,102,60);

DELETE FROM EMPLOYEES WHERE EMPLOYEE\_ID = 400;

SELECT \* FROM vw\_sales\_managers;

**Refreshing Materialized V?ews**

CREATE MATERIALIZED VIEW vw\_it\_programmers

BUILD IMMEDIATE

REFRESH FORCE

ENABLE QUERY REWRITE AS

SELECT \* FROM employees

WHERE job\_id = 'IT\_PROG'

ORDER BY department\_id;

**REFRESHING MANUALLY**

EXECUTE DBMS\_MVIEW.REFRESH('vw\_it\_programmers','F');

EXECUTE DBMS\_SNAPSHOT.REFRESH('vw\_it\_programmers','c');

EXECUTE DBMS\_MVIEW.REFRESH\_ALL;

select \* from employees where job\_id = 'IT\_PROG';

insert into employees values (333,'dfsadf','sdfsdf','asdf','34324',sysdate,'IT\_PROG',4444,NULL,102,70);

SELECT \* FROM vw\_it\_programmers;

DELETE FROM employees WHERE EMPLOYEE\_ID = 333;

DROP MATERIALIZED VIEW LOG ON employees;

CREATE MATERIALIZED VIEW LOG ON employees;

DROP MATERIALIZED VIEW deparment\_max\_salaries\_mv2;

**Scheduling Periodic Materialized View**

CREATE MATERIALIZED VIEW vw\_sales\_managers5

BUILD IMMEDIATE

REFRESH FORCE

START WITH to\_date('10-MAR-16','DD-MON-RR')

NEXT sysdate + 5

AS

SELECT \* FROM employees

WHERE job\_id = 'SA\_MAN'

ORDER BY department\_id;

--Retrieving data by using advanced subqueries

Select a.first\_name,a.last\_name,a.hire\_date,a.job\_id,b.department\_name,b.city from empvw80 a join

(select state\_province,city,department\_id,department\_name from departments join locations using (location\_id)) b

using (department\_id);

--Multiple-column- subqueries

SELECT first\_name, last\_name,manager\_id,department\_id from employees

where

department\_id in (select department\_id from employees where UPPER(first\_name) = 'LUIS')

and manager\_id in (select manager\_id from employees where UPPER(first\_name) = 'LUIS')

/

--pairwise subquery

SELECT first\_name, last\_name,manager\_id,department\_id from employees

where (department\_id, manager\_id) in (select department\_id, manager\_id from employees where UPPER(first\_name) = 'LUIS');

/

--scalar subquery

SELECT \* FROM employees;

--Example 1

SELECT employee\_id, first\_name,last\_name from EMPLOYEES

where department\_id = (select department\_id from employees where upper(first\_name) = 'LUIS');

--Example 2

SELECT employee\_id, first\_name,last\_name,

(SELECT department\_name FROM departments WHERE department\_id = e.department\_id ) AS department\_name

FROM EMPLOYEES e;

--Example 3

--For example; we know San Francisco's Postal Code and we want to find if an employee works in San Francisco or not ....

SELECT employee\_id, first\_name,last\_name,

(CASE WHEN location\_id = (select location\_id from locations where postal\_code = '99236') then 'San Franscisco'

ELSE 'Outside' END) COUNTRY FROM EMPLOYEES E NATURAL JOIN DEPARTMENTS D;

--Correlated Subquery

SELECT employee\_id, first\_name,last\_name,department\_id, salary FROM employees a

where salary = (SELECT max(salary) from employees b where b.department\_id = a.department\_id );

--Exist Operator

SELECT employee\_id, first\_name,last\_name,department\_id FROM employees a

WHERE EXISTS (SELECT 1,employee\_id FROM EMPLOYEES WHERE MANAGER\_ID = A.EMPLOYEE\_ID);

--Not Exist Operator

SELECT employee\_id, first\_name,last\_name,department\_id FROM employees a

WHERE NOT EXISTS (SELECT 1,employee\_id FROM EMPLOYEES WHERE MANAGER\_ID = A.EMPLOYEE\_ID);

SELECT department\_id, department\_name FROM departments d

WHERE not exists (select department\_id from employees where department\_id = d.department\_id);

SELECT department\_id, department\_name FROM departments d

WHERE department\_id not in (select department\_id from employees);

--Advanced Subqueries Using WITH Clause.

WITH MAX\_SALARIES AS

(

SELECT MAX(SALARY) maximum\_salary, trunc(AVG(salary)) average\_salary, department\_id

from EMPLOYEES GROUP BY DEPARTMENT\_ID)

SELECT employee\_id, maximum\_salary, average\_salary, department\_id FROM employees e

NATURAL JOIN MAX\_SALARIES;

--Using Recursive WITH Clause

WITH ALL\_MANAGERS(Employee,Manager,Department) AS

(

SELECT employee\_id, manager\_id, department\_id from employees

UNION ALL

SELECT all\_managers.employee, employees.manager\_id,all\_managers.department FROM all\_managers, employees

WHERE all\_managers.manager = employees.employee\_id

)

SELECT employee,manager,department from all\_managers order by employee;

--Inseting data by using subquery as a target

select \* from loc;

create table loc as select \* from locations;

create table con as select \* from countries;

drop table con;

INSERT INTO (SELECT l.location\_id, l.city, l.country\_id,c.country\_name,c.region\_id

FROM loc l

JOIN countries c

ON(l.country\_id = c.country\_id)

JOIN regions r on (r.region\_id = c.region\_id)

WHERE r.region\_name = 'Europe')

VALUES (3300, 'Cardiff', 'UK','Hello Omer',1);

create table departments\_temp as select \* from departments;

select \* from departments\_TEMP;

select \* from locations;

INSERT INTO

(SELECT d.department\_id, d.department\_name, d.manager\_id, d.location\_id

FROM departments\_temp d

WHERE location\_id =

(select location\_id from locations where city = 'Oxford')

WITH CHECK OPTION)

VALUES (300, 'Marketing', '145','2000');

--Correlated update

create table employees\_temp as select \* from employees;

select \* from employees\_temp;

update employees\_temp t set

(t.salary,t.commission\_pct) = (

select avg(salary),avg(e.commission\_pct) from employees e join departments d on (e.department\_id = d.department\_id)

where t.department\_id = d.department\_id

group by d.department\_id

);

--Correlated delete

delete from employees\_temp

where department\_id in

(select department\_id from departments d natural join locations l

where country\_id = 'UK')

--Manipulating data

create table departments\_temp as select \* from departments;

select \* from departments\_temp;

alter table departments\_temp modify (manager\_id number(6) default 100);

insert into departments\_temp (department\_id,department\_name,manager\_id,location\_id)

values (310,'Temp Department',default,2000);

update departments\_temp set manager\_id = default;

--Unconditional Insert All

create table employees\_history as select employee\_id,first\_name,last\_name,hire\_date from employees where 1=2;

create table salary\_history (employee\_id number(6),year number(4),month number(2),salary number(8,2),commission\_pct number(2,2));

insert all

into employees\_history values (employee\_id,first\_name,last\_name,hire\_date)

into salary\_history values (employee\_id,extract (year from sysdate),extract (month from sysdate),salary,commission\_pct)

select \* from employees where hire\_date > sysdate-365;

select \* from employees\_history;

select \* from salary\_history;

drop table IT\_PROGRAMMERS;

--Conditional Insert All

create table IT\_PROGRAMMERS as select \* from employees\_history where 1=2;

create table LIVING\_IN\_US as select \* from employees\_history where 1=2;

INSERT ALL

WHEN hire\_date > sysdate-365 THEN

INTO employees\_history values (employee\_id,first\_name,last\_name,hire\_date)

INTO salary\_history values (employee\_id,extract (year from sysdate),extract (month from sysdate),salary,commission\_pct)

WHEN job\_id = 'IT\_PROG' THEN

INTO IT\_PROGRAMMERS values (employee\_id,first\_name,last\_name,hire\_date)

WHEN department\_id in

(select department\_id from departments where location\_id in (select location\_id from locations where country\_id = 'US')) THEN

INTO LIVING\_IN\_US values (employee\_id,first\_name,last\_name,hire\_date)

SELECT \* FROM EMPLOYEES;

select \* from employees\_history;

select \* from salary\_history;

SELECT \* FROM IT\_PROGRAMMERS;

DELETE FROM SALARY\_HISTORY;

SELECT \* FROM LIVING\_IN\_US;

--Conditional INSERT FIRST

CREATE TABLE LOW\_SALARIES (EMPLOYEE\_ID NUMBER(6),DEPARTMENT\_ID NUMBER(6),SALARY NUMBER(8,2));

CREATE TABLE AVERAGE\_SALARIES AS SELECT \* FROM LOW\_SALARIES;

CREATE TABLE HIGH\_SALARIES AS SELECT \* FROM LOW\_SALARIES;

INSERT FIRST

WHEN SALARY <5000 THEN

INTO LOW\_SALARIES VALUES (EMPLOYEE\_ID,DEPARTMENT\_ID,SALARY)

WHEN SALARY BETWEEN 5000 AND 10000 THEN

INTO AVERAGE\_SALARIES VALUES (EMPLOYEE\_ID,DEPARTMENT\_ID,SALARY)

ELSE

INTO HIGH\_SALARIES VALUES (EMPLOYEE\_ID,DEPARTMENT\_ID,SALARY)

SELECT \* FROM EMPLOYEES;

SELECT \* FROM LOW\_SALARIES;

SELECT \* FROM AVERAGE\_SALARIES;

SELECT \* FROM HIGH\_SALARIES;

DROP TABLE LOW\_SALARIES;

DROP TABLE AVERAGE\_SALARIES;

DROP TABLE HIGH\_SALARIES;

--pivoting insert

create table job\_salaries (year NUMBER(4),month NUMBER(2),IT\_PROG NUMBER(8,2),SA\_MAN NUMBER(8,2),ST\_MAN NUMBER(8,2));

insert into job\_salaries VALUES(2015,8,

(select sum(salary+NVL(SALARY\*COMMISSION\_PCT,0)) from employees where job\_id = 'IT\_PROG'),

(select sum(salary+NVL(SALARY\*COMMISSION\_PCT,0)) from employees where job\_id = 'SA\_MAN'),

(select sum(salary+NVL(SALARY\*COMMISSION\_PCT,0)) from employees where job\_id = 'ST\_MAN'));

SELECT \* FROM JOB\_SALARIES;

CREATE TABLE JOB\_SAL (year NUMBER(4),month NUMBER(2),JOB\_ID VARCHAR2(20),TOTAL\_SALARY NUMBER(8,2));

insert all

into JOB\_SAL VALUES (YEAR,MONTH,'IT\_PROG',IT\_PROG)

into JOB\_SAL VALUES (YEAR,MONTH,'SA\_MAN',SA\_MAN)

into JOB\_SAL VALUES (YEAR,MONTH,'ST\_MAN',ST\_MAN)

SELECT \* FROM JOB\_SALARIES;

SELECT \* FROM JOB\_SAL;

drop table JOB\_SAL;

--Merge Operations

create table employees\_copy as

select employee\_id,first\_name,last\_name,department\_id,job\_id,salary from employees where job\_id = 'IT\_PROG';

SELECT \* FROM EMPLOYEES\_COPY;

MERGE INTO EMPLOYEES\_COPY C USING (SELECT \* FROM EMPLOYEES) E

ON (C.EMPLOYEE\_ID = E.EMPLOYEE\_ID)

WHEN MATCHED THEN

UPDATE SET

c.first\_name= e.first\_name,

c.last\_name = e.last\_name,

c.department\_id = e.department\_id,

c.job\_id = e.job\_id,

c.salary = e.salary

DELETE WHERE DEPARTMENT\_ID IS NULL

WHEN NOT MATCHED THEN

INSERT VALUES (E.employee\_id,E.first\_name,E.last\_name,E.department\_id,E.job\_id,E.salary);

SELECT \* FROM EMPLOYEES\_COPY;

SELECT \* FROM EMPLOYEES;

drop table employees\_copy;

---Flashback & purge

select \* from recyclebin;

ALTER TABLE EMPLOYEES\_COPY ENABLE ROW MOVEMENT;

delete from employees\_copy where salary > 5000;

commit;

select \* from employees\_copy;

flashback table employees\_copy to timestamp sysdate-1/1440;

select dbms\_flashback.get\_system\_change\_number scn from dual;

FLASHBACK TABLE EMPLOYEES\_COPY TO SCN 202381;

DROP TABLE EMPLOYEES\_COPY PURGE;

--Tracking Changes

--1

drop table employees\_copy;

--2

create table employees\_copy as select \* from employees;

select \* from employees\_copy ;

--3

update employees\_copy set salary = 1500

where employee\_id = 100;

commit;

--4

SELECT \* FROM employees\_copy

WHERE employee\_id= 100;

--5

SELECT \* FROM employees\_copy

AS OF sysdate -

WHERE employee\_id= 100;

--5.1

SELECT \* FROM employees\_copy

AS OF SCN 6167700

WHERE employee\_id= 100;

--CAN RUN IN SQL PLUS

SELECT DBMS\_FLASHBACK.GET\_SYSTEM\_CHANGE\_NUMBER FROM DUAL;

--6 VERSIONS

SELECT \* FROM V$TRANSACTION;

-----------------------------------------

--Flashback Versions Query

SELECT \* FROM employees\_copy

WHERE employee\_id= 100;

UPDATE employees\_copy SET salary = 2700

WHERE employee\_id = 100;

commit;

UPDATE employees\_copy SET salary = 3600

WHERE employee\_id = 100;

commit;

SELECT versions\_starttime,versions\_starttime, salary FROM employees\_copy

VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE

WHERE employee\_id= 100;

SELECT \* FROM employees\_copy

AS OF SCN 6167700

WHERE employee\_id= 100;

SELECT versions\_starttime,versions\_starttime, salary FROM employees\_copy

VERSIONS BETWEEN TIMESTAMP(sysdate - interval '13' minute) AND sysdate

WHERE employee\_id= 100;

--controlling schema objects

--adding constraint

alter table employees\_temp add constraint

temp\_cons unique (employee\_id);

alter table employees\_temp add constraint

temp\_cons2 unique (employee\_id,first\_name);

alter table employees\_temp add unique (phone\_number);

alter table employees\_temp modify job\_id constraint

not\_null\_job not null;

alter table employees\_temp modify first\_name not null;

alter table employees\_temp drop constraint temp\_cons;

alter table employees\_temp drop constraint SYS\_C0010501;

--on delete

ALTER TABLE salary\_history ADD CONSTRAINT sal\_emp\_fk

FOREIGN KEY (employee\_id)

REFERENCES employees\_temp(employee\_id) ON DELETE CASCADE;

SELECT \* FROM HIGH\_SALARIES;

ALTER TABLE HIGH\_SALARIES ADD CONSTRAINT hi\_emp\_fk

FOREIGN KEY (employee\_id)

REFERENCES employees\_temp(employee\_id) ON DELETE SET NULL;

--cascading constraints

select \* from all\_constraints where table\_name = 'EMPLOYEES\_TEMP';

alter table departments\_temp add constraint pk\_dep\_temp

primary key (department\_id );

ALTER TABLE employees\_temp ADD CONSTRAINT emp\_dp\_man\_fk

FOREIGN KEY (department\_id)

REFERENCES departments\_temp(department\_id);

create table departments\_temp as select \* from departments;

select \* from departments\_temp;

alter table departments\_temp drop column department\_id cascade constraints;

drop table departments\_temp;

create table departments\_temp as select \* from departments;

--renaming constraints

select \* from all\_constraints where table\_name = 'EMPLOYEES\_TEMP';

ALTER TABLE EMPLOYEES\_TEMP RENAME CONSTRAINT SYS\_C0010502 TO UQ\_EMP\_ID;

ALTER TABLE EMPLOYEES\_TEMP RENAME CONSTRAINT UQ\_EMP\_ID TO SYS\_C0010502;

--enabling and disabling constraints

alter table employees\_temp drop constraint SYS\_C0010502;

ALTER TABLE EMPLOYEES\_TEMP add constraint pk\_emp\_temp

primary key (employee\_id);

ALTER TABLE EMPLOYEES\_TEMP DISABLE CONSTRAINT SYS\_C0010502;

ALTER TABLE EMPLOYEES\_TEMP DISABLE CONSTRAINT SYS\_C0010502 CASCADE;

SELECT \* FROM HIGH\_SALARIES;

ALTER TABLE EMPLOYEES\_TEMP ENABLE CONSTRAINT SYS\_C0010502;

SELECT \* FROM HIGH\_SALARIES;

--Status of constraints

ALTER TABLE EMPLOYEES\_TEMP DROP CONSTRAINT SYS\_C0010502 CASCADE;

ALTER TABLE EMPLOYEES\_TEMP add constraint pk\_emp\_temp

primary key (employee\_id);

ALTER TABLE EMPLOYEES\_TEMP ENABLE NOVALIDATE PRIMARY KEY;

ALTER TABLE EMPLOYEES\_TEMP ENABLE NOVALIDATE CONSTRAINT EMP\_UQ2;

ALTER TABLE HIGH\_SALARIES DISABLE NOVALIDATE CONSTRAINT HI\_EMP\_FK;

--Deferring constraints

create table dep\_temp as select \* from departments;

ALTER TABLE DEP\_TEMP

ADD CONSTRAINT dep\_id\_pk

PRIMARY KEY (department\_id)

DEFERRABLE INITIALLY DEFERRED;

select \* from dep\_temp;

insert into DEP\_TEMP VALUES (10,'Temp Department',200,1700);

commit;

SET CONSTRAINTS dep\_id\_pk IMMEDIATE;

ALTER SESSION SET CONSTRAINTS = IMMEDIATE;

insert into DEP\_TEMP VALUES (10,'Temp Department',200,1700);

ALTER TABLE DEP\_TEMP DROP CONSTRAINT dep\_id\_pk;

ALTER TABLE DEP\_TEMP

ADD CONSTRAINT dep\_id\_pk

PRIMARY KEY (department\_id)

NOT DEFERRABLE;

SET CONSTRAINTS dep\_id\_pk IMMEDIATE;

SET CONSTRAINTS dep\_id\_pk DEFERRED;

--Temporary tables

create global temporary table shopping\_cart( id number, shopping\_date date)

on commit delete rows;

insert into shopping\_cart values(1,sysdate);

select \* from shopping\_cart;

commit;

select \* from shopping\_cart;

CREATE GLOBAL TEMPORARY TABLE sales\_managers

ON COMMIT PRESERVE ROWS AS

SELECT \* FROM employees

WHERE job\_id = 'SA\_MAN';

insert into sales\_managers (employee\_id,first\_name,last\_name,email,hire\_date,job\_id) values

('123','OMER','DAGASAN','oraclemaster@outlook.com',sysdate,'SA\_MAN');

insert into shopping\_cart values(1,sysdate);

select \* from sales\_managers;

commit;

select \* from sales\_managers;

--Privileges

select \* from system\_privilege\_map;

select \* from user\_sys\_privs;

select \* from session\_privs;

create user temp\_user identified by 123;

GRANT CREATE SESSION, CREATE TABLE, CREATE VIEW TO TEMP\_USER;

DROP USER TEMP\_USER;

-- ROLE

CREATE ROLE DEVELOPER;

GRANT CREATE TABLE, CREATE VIEW TO DEVELOPER;

GRANT DEVELOPER TO TEMP\_USER;

---CHANGE PASSWORD

password;

ALTER USER HR IDENTIFIED BY HR;

--Hierarchical Retrieval

--1

SELECT employee\_id, first\_name,last\_name, job\_id, manager\_id

FROM employees

START WITH employee\_id = 102

CONNECT BY PRIOR employee\_id = manager\_id ;

--2

SELECT employee\_id, first\_name,last\_name, job\_id, manager\_id

FROM employees

START WITH employee\_id = 102

CONNECT BY PRIOR manager\_id = employee\_id ;

--Changing Priors position

SELECT employee\_id, first\_name,last\_name, job\_id, manager\_id

FROM employees

START WITH employee\_id = 102

CONNECT BY employee\_id = PRIOR manager\_id;

---

SELECT LEVEL, employee\_id, first\_name,last\_name, job\_id, manager\_id

FROM employees where level = 2

START WITH employee\_id = 101

CONNECT BY manager\_id = PRIOR employee\_id ;

---Formatting Tree Structure with Using LEVEL and LPAD

SELECT LPAD(last\_name, LENGTH(last\_name)+(LEVEL\*2)-2,'> ')

AS employees\_hierarchy

FROM employees

START WITH employee\_id =101

CONNECT BY PRIOR employee\_id=manager\_id;

--Pruning Branches of a Tree

SELECT employee\_id,last\_name,manager\_id

FROM employees where employee\_id <> 108

START WITH employee\_id =101

CONNECT BY PRIOR employee\_id=manager\_id;

--Pruning Branches of a Tree

SELECT employee\_id,last\_name,manager\_id

FROM employees

START WITH employee\_id =101

CONNECT BY PRIOR employee\_id=manager\_id

AND employee\_id <> 108;

--Generating SQL Scripts

SELECT 'CREATE TABLE ' || table\_name || '\_backup ' || 'AS SELECT \* FROM '

|| table\_name AS

"Backup Scripts" FROM user\_tables;

SELECT 'CREATE TABLE ' || table\_name || '\_backup ' || 'AS SELECT \* FROM '

|| table\_name || ';' AS

"Backup Scripts" FROM user\_tables;

SELECT 'DROP TABLE ' || table\_name || ';'

FROM user\_tables WHERE table\_name like '%\_BACKUP%';

---------regular expressions-----------

--------------regexp\_like-------------

SELECT first\_name, last\_name

FROM employees

WHERE REGEXP\_LIKE (first\_name, '^Ste(v|ph)en$');

ALTER TABLE employees\_copy add constraint number\_format

CHECK ( REGEXP\_LIKE ( phone\_number, '^\d{3}.\d{3}.\d{4}$' ) ) novalidate;

select \* from employees\_copy where REGEXP\_LIKE ( phone\_number, '^\d{3}.\d{3}.\d{4}$' );