1.sql_topics

--basic sql statement

--relational operators

--single row function

--external table

--sorting data

--sqlloader

--group function

--analytical function

--set operators

--joins

--pseudo column

--constraint

--ddl

--dcl

--tcl

--dml

--views

--subquery

--sequence

--synonyms

--index

http://127.0.0.1:8080/apex/f?p=4550:11:113628644213605 0::NO:::

select from where having on--clauses

***---all columns**

select * from EMPLOYEES;

select columnname1,columnname2 / * from tablename

SeLect table_name from user_tables;

desc employees;

https://drive.google.com/file/d/1-L2yFuwwAPhc5pyBtk_kjPi8GagOijpD/view?usp=drivesdk

2.relational operators

relational operators:

>all--select first_name,salary from employees where salary >all (7000,17000);GREATER than greatest

<all--select first_name,salary from employees where salary <all (7000,17000);lesser than least

>any--select first_name,salary from employees where salary >any (7000,17000);greater than least

>any--select first_name,salary from employees where salary <any (7000,17000);lesser than greatest

like-select first_name,salary from employees where first_name like 'Lex'

notlike-select first_name,salary from employees where first_name not like 'Lex'

in-select first_name,salary from employees where salary in (4800,24000,17000);

notin-select first_name,salary from employees where salary not in (4800,24000,17000);

and-select first_name,salary from employees where first_name='David' and salary>5000

or-select first_name,salary from employees where first_name='David' or salary>5000)

>-select first_name,salary from employees where salary>5000

<-select first_name,salary from employees where salary<5000

<> not equal to-select first_name,salary from employees where salary<>5000

greater than>=-select first_name,salary from employees where salary>=5000

lesser than<=-select first_name,salary from employees where salary<=5000

is null-select first_name,salary from employees where department_id is null

is not null-select first_name,salary from employees where department_id is not null

between-select first_name,salary from employees where salary between 5000 and 10000;

not between-select first_name,salary from employees where salary not between 5000 and 10000;

3.DDL

ddl-data definition language

create

alter

(add

rename

modify

drop)

truncate

drop

datatypes

----

number(38)

char(2000)

varchar2(4000)

long(2gb)

clob(4gb)

blob(4gb)

date (to store date)

timestamp (date+time)

bfile   (filepath)

xmltype(to store xml data)

----

create

```
create table students
(sid number,
sname varchar2(30),
cid   number,
gender char(1)
dob     date);
```

```
INSERT all
  INTO students (sid,sname,cid,gender,dob) VALUES
(001,'saran',101,'m','04-sep-1999')
```

```
  INTO students (sid,sname,cid,gender,dob) VALUES
(002,'sarath',102,'m','04-sep-2000')
  INTO students (sid,sname,cid,gender,dob) VALUES
(003,'deepak',103,'m','04-sep-1998')
  INTO students (sid,sname,cid,gender,dob) VALUES
(004,'rakesh',104,'m','04-sep-1997')
  INTO students (sid,sname,cid,gender,dob) VALUES
(005,'rahul',105,'m','04-sep-1996')
select * from dual;
--------------------------------------------------------------------------------
--------------------
alter
----------
add
---

alter table students add faculty_name varchar2(30);

modify
------

alter table students modify faculty_name varchar2(40);
```

rename

----

alter table students rename column faculty_name to teacher_name;

drop

----

alter table students drop column teacher_name;

truncate: delete all data only inside the table

truncate table students;

drop: delete full structure of the table from database

drop table students;

4.DML

dml-data manuplation languages

insert

update

delete

insert-insert data into table

```
INSERT INTO students (sid,sname,cid,gender,dob) VALUES
(001,'saran',101,'m','04-sep-1999');
```

---

insert all: insert multiple rows of data into the table

--

```
INSERT all
  INTO students (sid,sname,cid,gender,dob) VALUES
(001,'saran',101,'m','04-sep-1999')
  INTO students (sid,sname,cid,gender,dob) VALUES
(002,'sarath',102,'m','04-sep-2000')
```

```
  INTO students (sid,sname,cid,gender,dob) VALUES
(003,'deepak',103,'m','04-sep-1998')
  INTO students (sid,sname,cid,gender,dob) VALUES
(004,'rakesh',104,'m','04-sep-1997')
  INTO students (sid,sname,cid,gender,dob) VALUES
(005,'rahul',105,'m','04-sep-1996')
select * from dual;
```

----

update update field in the table or database

```
update students set sid=6
where sid=5
```

```
update table set column name=value
where column name = value
```

----

delete: delete particular row from table

```
delete from students where sid=6;
```

5.TCL

tcl transaction control unit

commit - to save all pending changes to permenant

rollback-to discard all pending record

savepoint-its a marker

t1

1

2

commit;

3

4

savepointx;

5

6

rollback to x;

6.CONSTRAINTS

constraints-Constraints are used to limit the type of data that can go into a table

NOT NULL - Ensures that a column cannot have a NULL value

UNIQUE - Ensures that all values in a column are different

PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table

FOREIGN KEY - Prevents actions that would destroy links between tables

CHECK - Ensures that the values in a column satisfies a specific condition

DEFAULT - Sets a default value for a column if no value is specified

==

```
create table emp

(

emp_id number(10),

constraint cons_empid_pk primary key(emp_id),

name varchar2(20) not null,
institute varchar2(10) default 'besant',
email  varchar2(30) unique,
gender varchar2(10),
check(gender in ('m','f'))
);


insert into emp(emp_id,name,email,gender)
values(1,'saran',null,'m')
insert into emp(emp_id,name,email,gender)
values(2,'rahul',null,'m')
```

==
==================================

foreign key

```
create table department
(
dept_id number,
CONSTRAINT department_pk PRIMARY KEY (dept_id),
dept_name varchar2(30) not null);
```

```
insert into department(dept_id,dept_name) values
(10,'aws')


create table emp

(

emp_id number(10) NOT NULL,

constraint cons_empid_pk primary key(emp_id),

name varchar2(20) not null,

institute varchar2(10) default 'besant',

email  varchar2(30) unique,

gender varchar2(10),

check(gender in ('m','f')),

dept_id number,

 CONSTRAINT fk_dept

    FOREIGN KEY (dept_id)

    REFERENCES department(dept_id)

);


insert into emp(emp_id,name,email,gender,dept_id)
values(1,'saran',null,'m',10);
```

select e.*,d.* from emp e,department d where e.dept_id=d.dept_id;

desc emp

7.JOINS

joins

create table emp as select*from employees where rownum<=5

joins are used to join two or more tables on the table itself

equi join/inner join

self join

outerjoin

- left outer join
- right outer join
- full outer join

1)inner join

*it is also cross join but a cross join with equi condition

```
select
e.employee_id,first_name,d.department_id,department_na
me from employees e,departments d where
e.department_id=d.department_id
```

```
select
e.employee_id,first_name,d.department_id,d.department_
name from employees e,departments d
where e.department_id=d.department_id
```

```
select
e.employee_id,first_name,d.department_id,department_na
me from employees e inner join departments d
on e.department_id=d.department_id where
d.department_id in (10,20,30)
```

```
select
e.employee_id,first_name,d.department_id,department_na
me,l.location_id from employees e,departments d,locations
l where e.department_id=d.department_id
and d.location_id=l.location_id
```

----

2) self join

self join is a join which join the table itself

different alias name are taken for same table

```
select
e1.employee_id,e1.first_name,e2.employee_id,e2.first_nam
e from employees e1,employees e2
where e1.manager_id=e2.employee_id and e1.salary >
e2.salary
```

----

3)left outer join

it will give all the values from left side and matched values from right side

```
select
e.employee_id,first_name,d.department_id,d.department_
name from employees e left outer join departments d
on e.department_id=d.department_id
```

right outer join

it will give all the values from right side and matched values from left side

```
select
e.employee_id,first_name,d.department_id,d.department_
name from employees e right outer join departments d
on e.department_id=d.department_id
```

full outer join

```
select
e.employee_id,first_name,d.department_id,d.department_
name from employees e full outer join departments d
on e.department_id=d.department_id
```

8.SEQUENCE&SYNONYM

sequence: user defined schema bound object that generates a sequence numeric values

according to the specification with which the sequence was created

```
create sequence seq
start with 1
```

increment by 1

maxvalue 10;

alter sequence seq

increment by 2

maxvalue 30;

select seq.nextval from dual;

select seq.currval from dual;

--

synonyms:alternate name fro an object (it will generate duplicate name)

synonyms can create for synonyms

create synonym syn for employees;

select * from syn;

create synonym syn1 for syn;

select * from syn1;

select * from user_synonyms;

9.GROUP FUNCTION & SORTING DATA

group function

group function operates over number of values with in column returns a single value

sum()

max()

min()

count()

avg)()

sum():it gives total from the number data type column

select sum(salary) from employees;

----

max():it will give max among the salary

select max(salary) from employees;

----

min():it will gives min among salary

select min(salary) from employees;

----

count():its gives number of record,it will count only not null values

select count(salary) from employees;

----

sorting data:

order by

group by

orderby:desc,asc

select * from employees order by first_name asc; desc;

groupby: this clause is used to divide similar data item into set of logical group

whatever column is select that should be in groupby

select department_id,sum(salary) from employees group by department_id;

instead of where clause written having after groupby

select department_id,sum(salary) from employees group by department_id having sum(salary)>5000;

10.VIEWS

views

views is a virtual table is a logically represent subset of data from one or more tables

views store only the query

views are not store in the database

types of views

-simple view

-complex view

simple view

```
create or replace view v1
as
select*from employees;

select * from v1;
```

advantages

to make complex query look simple to restrict data access

if a view created on single base table is called view

--complex view

*a view cannot perform dml operation

*a view created with multiple base table are called complex view

```
create or replace view v1
as
select e.first_name,e.department_id,d.department_name
from employees e,departments d where
e.department_id=d.department_id;
```

11.INDEX

index- is used to improve the performance of the query

b-tree/bitmap

composite

function based

unique

```
select * from user_ind_columns;
```

oracle keyword --select from syntactic checking

semantic checking --table column

----to reduce cost of the query

index is database object which is used to fetch data very fast from the database

this process automatically improves performance of query

===

======================================

--b-tree index

select first_name,salary from employees

where first_name='Ellen';

create index ind on employees(first_name asc);

======================

--composite index-- index created with multiple column select first_name,salary from employees

where first_name='Steven' and last_name='King';

create index idx3 on employees(first_name,last_name)

==
====================================

--function based idex--its create extension of btree index we can do function function inside the index

select first_name,salary from employees where length(first_name)=4;

create index idx4

on employees(length(first_name));

==
======================================

--unique index--its never allow dulpicate

create table t1(a number);

12.EXTERNAL TABLES

external tables allow oracle to query data that is stored outside the database in flatfile

the oracle_loader driver can be used to access any data stored in any format

that can be loaded by sql*loader

no dml can be perform on external table

```
C:\Users\Saravanan>sqlplus sys as sysdba

SQL*Plus: Release 10.2.0.1.0 - Production on Tue Jan 17
14:32:04 2023

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

Enter password:admin

Connected to:
Oracle Database 10g Express Edition Release 10.2.0.1.0 -
Production

SQL> grant create any directory to hr;
```

```
Grant succeeded.

SQL> grant execute on utl_file to hr;


Grant succeeded.


SQL> grant read,write on directory new to hr;


Grant succeeded.
```

```
------------------
```

32767 number of character we can write in a line

```
------------------------
```

```
create or replace directory new as 'D:\';



create directory path as 'D:\'


create table students_ext (
  student_code varchar2(5),
  student_name varchar2(50),
  student_language  varchar2(50)
)
```

```
organization external (
 type oracle_loader
 default directory path
 access parameters (
   records delimited by newline
   fields terminated by ','
   missing field values are null
   (
     student_code char(5),
     student_name char(50),
     student_language  char(50)
)
)
location ('students1.txt','students2.txt')
)
parallel 5
reject limit unlimited;
```

13.SINGLE ROW FUNCTION

single row function

----

case manuplation function

----

upper

lower

initcap

```
select
first_name,upper(first_name),lower(first_name),initcap(first
_name) from employees;
```

upper=all letter are capital

lower=all letter are smaller

initcap=first letter only capital

--
----

general function

----

greatest

it gives greatest value in select statement

```
select greatest (9,8,10,12) from dual;
```

least

it gives least value in select statement

select least (9,8,10,12) from dual

----

control statement

----

decode

decode compares the expression to each search value one by one if expression is equal to a search then corresponding result is returned by the oracle database

select department_id,decode(department_id,90,'HR',60,'sales','others') from employees

case

case is not an expression not a statement

select department_id,

case when department_id=90 then 'HR' when department_id=60 then 'sales' end from employees;

CONCATENATION:ADDS TWO OR MORE STRING TOGETHER

SELECT FIRST_NAME||LAST_NAME||' '||SALARY FROM EMPLOYEES

--

date function

----

dd-mon-yy its give computer today date

select sysdate from dual;

month between

select months_between(sysdate,'21-jan-22') from dual;

select trunc(months_between(sysdate,'21-jan-22')) from dual;

select trunc(months_between('21-jan-23','21-jan-22')) from dual;

add date

select sysdate+10 from dual;

add mon

select add_months(sysdate,10) from dual;

next day

select next_day(sysdate,'monday') from dual;

lastday(we can find last date of the month)

select last_day('2-aug-23') from dual;

--

character manuplation function

----

substr(): it gives a part of string

select substr('besant',2,4) from dual;

select first_name,substr(first_name,1,3) from employees;

instr():it will give position of string or characters

select instr('besant','a',1,1) from dual;

translate:it will replace character by character

select translate('welcome','em','xy') from dual;

select translate('welcome','em','x') from dual;

replace: it will take as a complete string

select replace ('welcome','come','sys') from dual;

length: it gives length of string

select length('welcome') from dual;

concat: join 2string value

select concat ('hello','world') from dual;

select concat(first_name,last_name) as name from employees

----

null function

----

nvl:if the 1st argument null it give 2nd argument

select nvl(null,3) from dual;

select nvl(2,3) from dual;

select first_name,nvl(department_id,1000) from employees;

select salary,salary+nvl(COMMISSION_PCT,0) from employees;

nvl2 it accept 3argument if the 1st argument null give 3rd value

select nvl2(1,2,3) from dual

select nvl2(null,2,3) from dualelse its return 1st value

select nullif(3,3) from dual

select nullif(3,4) from dual

number function

----

trunc:round the value will eliminate the decimal point

select trunc(1234.56) from dual;

select trunc(1234.5678,2) from dual;

round:

select round(12345.678) from dual;

select round(12345.678,2) from dual;

abs:it return absolute value of number

select abs(-145.35) from dual

select abs(145.35) from dual

sqrt:it will sqrt of arguments

select sqrt(16) from dual

=========================

Conversion Functions

****

select to_char(hire_date,'MM/YYYY/DD') from employees

select to_date('1995-OCT-05','YYYY-MON-DD') from dual;

14.ANALATICS FUNCTIONS

rank()

dense_rank()

lead()

lag()

rank():any duplicate values fount it skips nextvalue

select first_name,salary,rank() over(order by salary desc) from employees;

----

dense_rank():any duplicate values fount it will not skips nextvalue

select first_name,salary,dense_rank() over(order by salary desc) from employees;

----

lead(): used to compare values of the current row with previous and nextrow values

select first_name,lead(first_name,1,'x')over(order by first_name desc) from employees;

----

lag(): it will display current values with prior values

select first_name,lead(first_name,1,'x')over(order by first_name desc) from employees;

15.SUBQUERY

subquery

a query inside a query is called a subquery

nested subquery

corelated subquery

inlineview subquery

scalar subquery

1)inline view

subquery written in the from clause

select * from (select e.*,rownum as rn from employees e)
where rownum=1

2)scalar subquery

subquery written in the select clause

select 2+ (select 3+4 from dual) from dual;

select e.first_name,(select max(salary) from employees)
from employees e WHERE SALARY=(SELECT MAX(SALARY)
FROM EMPLOYEES);

3)corelated subquery

if the inner query depends on outer query is called
corelated subquery

```
SELECT * FROM EMPLOYEES E WHERE E.DEPARTMENT_ID IN
(SELECT D.DEPARTMENT_ID FROM DEPARTMENTS
D,EMPLOYEES E WHERE
E.DEPARTMENT_ID=D.DEPARTMENT_ID)
```

4) nested subquery

```
SELECT * FROM EMPLOYEES WHERE SALARY=(SELECT
MAX(SALARY) FROM EMPLOYEES);
```