INTRODUCTION TO ORACLE DATABASE SQL COMMANDS

STRUCTURED QUERY LANGUAGE (SQL)

SQL can communicate with Oracle Database Server. It has the following advantages

- Effective
- Easy to learn and use
- Functionality complete (with SQL you can define, retrieve and manipulate data in tables)

SQL statements

1. Data Manipulation Language (DML)
    - Select
    - Insert
    - Update
    - Delete
    - Merge

2. Data Definition Language (DDL)
    - Create
    - Alter
    - Drop
    - Rename
    - Truncate
    - Comment

3. Data Control Language (DCL)
    - Grant
    - Revoke

4. Transaction Control Language (TCL)
    - Commit
    - Rollback
    - Savepoint

Working with Oracle Database

Oracle Database provides an organized mechanism for storing, managing, and retrieving information.Tables are the basic storage structure for holding business data.

Creating Tables

You create tables with the SQL CREATE TABLE statement. With Oracle Database , you have two options for creating tables.

- Use the graphical interface that generates the SQL statement
- Enter the CREATE TABLE statement in the SQL Workshop tool

When creating tables, you must provide:

- Table name

- Column name(s)
- Data types for each column

Guidelines for creating tables:

- Table and column naming rules

  - Must start with a letter, which is followed by a sequence of letters, numbers,_,#,0r $

  - Must be 1 to 30 characters long

  - Must not be an oracle server reserved word

- **Oracle data types**

  VARCHAR2(n): Variable length charter string up to n characters

  CHAR(n): Fixed length charter string of n characters

  NUMBER(n): Integer number of up to n digits

  NUMBER(precision, scale): Fixed-point decimal number. “precision” is the total number of digits; “scale” is the number of digits to the right of the decimal point. The decimal point is not counted.

  NUMBER: Floating-point decimal number

  DATE: DD-MON-YY (or YYYY) HH:MM:SS A.M. (or P.M.) form date-time.

  LONG: Variable-length character string up to 2 GB. Only one long-type column in a table. (1 byte ASCII char)

  NCHAR: LONG for international character sets (2-byte per character)

  CLOB: Single-byte ASCII character data up to 4 GB

  NCLOB: 2-byte CLOB

  BLOB: Binary data (e.g., program, image, or sound) of up to 4 GB

BFILE: Reference to a binary file that is external to the database (OS file)

RAW(size) or LONG_RAW: raw binary data

ROWID: Unique row address in hexadecimal format

You can also set up constraints on your columns to control the data in them.

CREATING DATABASES BY USE OF SQL

1. **Creating a database user by logging in a system user**

```
SQL >  connect system/tuk123;           (or the password )

 SQL > create user student2020
          identified by tuk123;

SQL > grant create session to student2020

SQL > grant dba to student2020;
```

2. **Connect as user student as transact**

```
 SQL >  connect student2020/tuk123;

          SQL >  Create table employees
          (
          employee_id                 number(7)  not null,
          first_name                   varchar2(20),
          last_name                   varchar2(20),
          cellphone                   varchar2(12),
          email                       varchar2(20),
          hire_date                  date,
          job_id                      varchar2(5),
          salary                      number(12,2),
          manager_id                   number(6),
          department_id                number(4)
          );
```

```
SQL > Create table jobs_grade
 (
jobs_id                        varchar2(5) not null,
job_title                      varchar2(20),
min_salary                      number(8),
max_salary                       number(8)
);


SQL >  Create table department
 (
 department_id                   number(4)  not null,
 department_name                  varchar2(20),
 manager_id                        number(6),
 location_id                       number(4)
);
```

Adding data into table employees

```
SQL >  insert into employees
values(1000,'Simon', 'Otieno','0722456789','otieno@yahoo.com','01-jan-90','5500',32000,
5000,10);

SQL >  insert into employees
values(1001,'Alice', 'Mwangi','0720766659','alice@yahoo.com','02-feb-80’,'5600',42000, 5000,
10);
```

Adding data into jobs_grade

```
SQL > insert into jobs_grade
values('5500',' Accountant',10000,100000);

SQL >  insert into  jobs_grade
values('5600',' Database specialist',30000,150000);
```

Adding data into department

```
SQL > insert into department
values(10,'Finance',5000,22);

SQL > insert into department
values(20,'Human Resources',5100,41);
```

DESCRIBE

List the table structure

examples

```
desc employees;
desc  jobs_grade;
desc  department;
```

WRITING SQL STATEMENTS

i. SQL statements are not case sensitive
ii. SQL can be entered on many lines
iii. Keywords cannot be split across lines
iv. Clauses are usually placed on separate lines for readability and ease of editing
v. Indents make it more readable
vi. Keywords may be entered in caps and all others in lowercase

SELECT – retrieving data using SQL select statement

This is used to view, query or report on data from tables. A select statement retrieves information from the database. With select statement you can use the following capabilities

1. *Projection- choose columns/fields from a table through a query.*
2. *Selection- choose rows in a table*
3. *Joining – bring together data that is stored in different tables by specifying the link between them.*

examples

```
select * from employees;
select * from jobs_grade;
select * from department;
select employee_id,first_name,last_name, email , job_id,    salary from employees;
select jobs_id,job_title ,min_salary, max_salary  from jobs_grade;
select department_id , department_name ,manager_id ,location_id from  department;
```

Defining a Column/field as ALIAS (renaming fields with the select statement)

```
select employee_id “Employee ID”,first_name “First Name” ,last_name “Last Name”, email “Email” ,
job_id “Job ID”,salary “ Monthly Pay “ from employees;
```

ARITHMETIC EXPRESSIONS

Operator	Description
+	**Add**
-	**Subtract**
*	**Multiply**
/	**Divide**

examples;

```
Select employee_id, first_name,last_name, salary, salary + 3000 from employees;

Select employee_id, first_name,last_name, salary, 12* salary + 700 from employees;

Select employee_id, first_name,last_name, salary, 12* (salary + 700) from employees;

Select employee_id, first_name,last_name, salary, 12* (salary - 2000) from employees;
```

NB: Use of BODMAS applies in arithmetic expressions

DUPLICATE ROWS /RECORDS;

example

```
select distinct department_id from employees;
```

RESTRICTING AND SORTING DATA WITH SELECT STATEMENT

use of the where clause;

Comparison conditions

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to
Between … And	Between two values
IN (set)	Match any of the list
Like	Match a character pattern
IS Null	is a null value

examples

```
select employee_id “Employee ID”,first_name “First Name” ,last_name “Last Name”, email
“Email” , job_id “Job ID”,salary “ Monthly Pay “ from employees
where employee_id=1000;

select employee_id “Employee ID”,first_name “First Name” ,last_name “Last Name”, email
“Email” , job_id “Job ID”,salary “ Monthly Pay “ from employees
where salary > 10000;

select employee_id “Employee ID”,first_name “First Name” ,last_name “Last Name”, email
“Email” , job_id “Job ID”,salary “ Monthly Pay “ from employees
where salary  in (10000,20000,30000);

select  last_name, salary
from employees
where salary <= 10000
```

```
select  last_name, salary
from employees
where salary between 3000 and 15000;
```

```
select employee_id,last_name, salary, manager_id
from employees
where manager_id in (100,101,201);
```

```
select employee_id, first_name,last_name
from employees
where first_name like 'S%';
```

```
select employee_id,last_name, salary, manager_id
from employees
where manager_id is null;
```

Logical Conditions

Operator	Meaning
AND	**Returns true is both are true**
OR	**Returns true if one is true**
NOT	**Return true if condition is false**

Examples

```
select employee_id,last_name
from employees
where salary >= 10000 and manager_id=5000;
```

```
select employee_id,last_name
from employees
where department_id not in (90,60,30);
```

USING THE ORDER BY CLAUSE

```
select last_name,job_id, department_id
from employees
order by hire_date desc;
```

```
select last_name,job_id, department_id
from employees
order by hire_date asc;
```

NB : ascending or descending;

UPDATE COMMAND

This is used to update data in given tables *examples*

```
update employees
set salary= 50000
where employee_id=1001;

update employees;
set last_name='Opiyo';
where employee_id=1000;

update employees
set department_id=70
where employee_id=1001;
```

DELETE COMMAND

Used to delete or remove records from tables

examples

```
delete from employees
where employee_id=1000;
```

ROLLBACK

Undo some transactions; used with data manipulation language commands

example

```
rollback;
```

COMMIT

ensures that records are permanently saved. used with data manipulation language commands

example

```
commit;
```

CREATING A COPY OF A TABLE

examples

```
create table employees2
as select * from employees;

create table employees3
as select * from employees;

create table jobs_grade2
as select * from jobs_grade;

create table jobs_grade3
as select * from jobs_grade;
```

```
create table department2
as select * from department;

create table department3
as select * from department;
```

CREATING A TABLE BU USING A SUBQUERY

```
create table department10
as select employee_id,last_name,first_name,salary, salary*(120/100) NewSalary, hire_date
from employee where department_id=10;
```

TRUNCATE COMMAND

Remove all rows but leave the table structure intact;

example

```
truncate employees3;
```

DROP TABLE COMMAND

i. All data and structure is deleted
ii. Pending transactions are not committed
iii. All indexes are dropped
iv. All constraints are dropped
v. You cannot rollback drop table

examples

```
drop table jobs_grade3;

drop table department3;

drop table employees3;
```

USING SINGLE ROW FUNCTIONS TO CUSTOMIZE OUTPUT

Character functions

i. lower
ii. upper
iii. initcap
iv. concat
v. substr
vi. length
vii. trim
e.tc

Examples

select lower(last_name), upper(last_name) from employees;

select initcap(first_name) from employees;

Number functions

i. Round – round value to specified decimal
ii. Trunc – Truncates value to specified decimal
iii. Mod – returns remainder of division

Examples

select round(45.92356,2) from dual;

select trunc(45.92356,2) from dual;

select mod(1600,300) from dual;

Working with dates

Select last_name , hire_date from employees
where hire_date < '01-feb-88';

select sysdate from dual;

select sysdate + 7 from dual;

select systdate – 7 from dual;

REPORTING GROUPED DATA

TYPE OF GROUPED FUNCTIONS

1. avg
2. count
3. max
4. min
5. stddev
6. sum
7. variance

etc

examples

select max(salary), min(salary), sum(salary),avg(salary) from employees;

select min(hire_date), max(hire_date) from employees;

select count(*) from employees;

```
select count(salary) from employees where department _id in (10,20,30,40);

select count(distinct department_id) from employees;

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

select avg(salary) from employees group by department_id;

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

HAVING CLAUSE

```
select department_id,max(salary) from employees group by
department_id
having max(salary) > 10000 ;
```

DISPLAYING DATA FROM MULTIPLE TABLES

This is achieved using

1. *cross joins*
2. *natural joins*
3. *using clause*
4. *full outer joins*
5. *arbitrary joins*

examples

```
select employees.employee_id, employees.last_name,department_id from employees join
departments using (department_id)
```

INCLUDING CONSTRAINTS

1. *Enforce rules at table level*
2. *Prevent deletion of a table if there are dependencies*
3. *Following constraints are valid*
   - *Not null*
   - *Unique*
   - *Primary key*
   - *Foreign Key*
   - *Check*

example

```
create table employees5
(
employee_id number(6) constraint employees5_id_pk primary key,
first_name varchar2(20)
);
```

A. Not null constraint ensures that the column contains no null or empty values

B. unique key constraint – requires that every value must be unique

example

```
create table employees31
(
employee_id number(6),
last_name varchar2(20) not null,
email varchar2(20),
salary number(10,2),
hire_date date  not null,
constraint emp_email_uk unique(email)
);
```

C. Primary key – Constraint creates a primary key for the table. Only one primary key can be created for each table.

D. Foreign key – or referential integrity constraint designates a column or combination of columns as a foreign key and establishes a relationship between a primary key or a unique key in the same table or different table

example

Creating foreign key constraint

```
Create table department50
(
Department_id number(5) constraint department_id_pk primary key,
Department_name vachar2(20)
)


create table employees51
(
employee_id number(6)  constraint emp_id_pk primary key,
last_name varchar2(20) not null,
first_name varchar2(20),
salary number(10,2),
department_id number(4),
 constraint  emp_dept_fk foreign key (department_id) references department50(department_id)
)
```

NB: the table department must exist with primary key on department_id

E. Check constraint

```
create table employees35
(
employee_is number(6),
first_name varchar2(20),
last_name varchar2(20) not null,
email varchar2(20),
salary number(10,2) constraint  emp_salary_min_check check (salary >0)
);
```

Integrity constraint error

When you have constraints in place on columns, an error is returned if you try to violate the constraint rule.

ALTER TABLE STATEMENTS

This is used to

i. Add a new column to a table
ii. Modify an existing column
iii. Define default value for a new column
iv. Drop a column from a table

example

```
alter table jobs_grade
add column effective_date date;

alter table employees
add constraint emp_id_pk primary key;

alter table jobs_grade1
drop column jobs_title;

alter table employees10
drop constraint emp_dept_fk;
```

1.Adding a new Column

```
alter table employees11
  add  net_pay number(12,2);
```

2.Rename a column

```
alter table employees11
  rename column tax to deduction;
```

3.Modify a column

```
alter table employees11
  modify tax number(10,2);
```

4 **Drop a Column from a table**

```
alter table employees11
  drop column deduction;
```

DATA OBJECTS

OBJECT	DESCRIPTION
Table	Basic unit of storage
View	Logically represents subsets of data from one or more tables
Sequence	Generate numeric values
Index	Improves the performance of some queries
Synonyms	Gives alternative names to objects

What is a view ?

A view is a logical table based on a table or another view

Advantages of a view

1. To restrict data access
2. To make complex queries easy
3. To provide data independence
4. To present different views of the same data

Creating a View

create view empvu80
*as select * from employees;*

Modifying a view

can be done by create or replace view

Removing a view

views can be removed by drop command
e.g *drop view empvu80*

Sequences

A sequence is a database object that creates integer values. You can create sequences and use them to generate numbers

create sequence sequence1
increment by 1
start with 2
maxvalue 100
nocycle;

```
SQL> select SEQUENCE_NAME,min_value,max_value,increment_by,last_number from user_sequences;
```

Indexes

Indexes are database objects that you can create to improve on the performance of some queries

create index emp_last_name_idx
on employees(last_name)

Removing an Index

drop index index_name e.g *drop emp_last_name_idx*

Synonyms

Synonyms are database objects that enables you to call a table by another name

Examples

create synonym muchiri for employees;

DATA DICTIONARY

This is a collection of tables and views in oracle database created and maintained by oracle server and contains information about a database. It is an important tool for all users from end users to application developers and database administrators.

DATA DICTIONARY STRUCTURE

1. *USER – user's views*
2. *ALL – expanded user view*
3. *DBA – database admin view*
4. *V$ - performance related data*

NB :To use commands in this category you must be logged in as DBA or System user

examples
describe dictionary;

*select * from dictionary;*

desc user_tables;

select table_name from user_tables;

desc user_constraints;

*select * from user_constraints*
where table_name='employees';