CREATE_TABLE

```
create table department(
                dep_id number(5) primary key,
                department_name varchar2(30)
                );
```

CREATE_TABLE BY ANOTHER TABLE

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

CREATE_TABLE_WITH_FOREIGN_KEY

```
create table employee(
           emp_id number primary key,
           emp_name varchar2(30),
           mobile varchar2(15) unique,
           salary number(7,2) check(salary > 10000),
           joining_date date default SYSDATE,
           country varchar2(30) default 'BD',
           dep_id number(5), constraint dep_emp_fk FOREIGN KEY (dep_id)
REFERENCES department(dep_id)
```

SHOW TABLE STRUCTRES

```
describe department;
```

CREATE INSERT_TRIGGER:

```
CREATE or REPLACE TRIGGER employee_after_insert AFTER INSERT ON employee
FOR EACH ROW
DECLARE
BEGIN
insert into employee_backup values(:new.emp_id, :new.emp_name, :new.mobile,
:new.salary, :new.joining_date, :new.country, :new.dep_id);
DBMS_OUTPUT.PUT_LINE('Record Successfully Inserted Into employee_backup Table');
END;
/
```

CREATE UPDATE_TRIGGER

```
CREATE or REPLACE TRIGGER employee_after_update AFTER UPDATE on employee
FOR EACH ROW
DECLARE
BEGIN
update employee_backup
set emp_name = :new.emp_name, mobile = :new.mobile, salary = :new.salary,
joining_date = :new.joining_date, country = :new.country, dep_id = :new.dep_id
where emp_id = :old.emp_id;
DBMS_OUTPUT.PUT_LINE('Record Successfully Updated Into employee_backup Table');
END;
/
```

CREATE DELETE_TRIGGER

```
CREATE or REPLACE TRIGGER employee_after_delete
AFTER DELETE ON employee
FOR EACH ROW
DECLARE
BEGIN
Delete employee_backup
where emp_id = :old.emp_id;
DBMS_OUTPUT.PUT_LINE('Record Successfully Deleted From employee_backup Table');
END;
/
```

CREATE SEQUENCE

```
create sequence employee
    start with 8
    increment by 2
    maxvalue 10000
    nocycle
    nocache;
```

SHOW CREATED SEQUENCE LIST

```
select sequence_name from user_sequences;
```

SHOW INDEX:

```
select index_name from user_indexes;
```

CREATE INDEX

```
CREATE INDEX emp_INDX ON emp (empno, deptno) TABLESPACE index_tbs;
```

OR

```
create index dep_name_idx on department(department_name);
```

CREATE INSERT PROCEDURE

```
CREATE OR REPLACE PROCEDURE insertDepertment(
      p_id IN DEPARTMENT.dep_id%TYPE,
      p_name IN DEPARTMENT.department_name%TYPE)
      IS
      BEGIN
      INSERT INTO DEPARTMENT(dep_id, department_name)
      VALUES(p_id, p_name);
      END;
      /
```

CALL/DATA INSERT BY PROCEDURE

```
BEGIN
   insertDepertment(dep_id_seq.nextval,'ADMIN');
END;
```

CREATE UPDATE PROCEDURE

```
CREATE OR REPLACE PROCEDURE updateCustomer(
        p_id IN CUSTOMER.id%TYPE,
        p_name IN CUSTOMER.name%TYPE)
IS
BEGIN
  update CUSTOMER set name = p_name where id = p_id;
END;
/
```

CALL/DATA UPDATE BY PROCEDURE

```
Begin
updateCustomer(100, 'Mr. Mahbub');
end;
```

CREATE DELETE PROCEDURE

```
CREATE OR REPLACE PROCEDURE deleteCustomer(
        p_id IN CUSTOMER.id%TYPE)
IS
BEGIN
delete from CUSTOMER where id = p_id;
END;
```

CALL/DATA DELETE BY PROCEDURE

```
Begin
deleteCustomer(100);
end;
```

CREATE VIEW

```
create view depv as select dep_id, department_name from department;
```

SHOW VIEW LIST

```
select view_name from user_viewes;
```

ADD A COLUMN

```
ALTER TABLE employee ADD(email VARCHAR2(30) unique);
```

CREATE SYNONYM

```
CREATE SYNONYM emp FOR SCOTT.EMP;
```