```
/*PL/SQL Program to find factorial of a Number.*/
DECLARE
 num number;
 res int default 1;
BEGIN
 num:=:num;
 while num >= 1 loop
  res := res * num;
  num := num - 1;
 END LOOP;
 DBMS_OUTPUT.PUT_LINE('Factorial is '||res);
END;
```

```
Factorial is 120
```

Statement processed.

Find the shipment information (supplier number, supplier name, part number, part name, quantity) for those parts having quantity less than 150

SELECT * FROM PART;

P_NUM	P_NAME	COLOR	WEIGHT	CITY
160301	PART_P1	NAVY BLUE	40	PUNE
160302	PART_P2	DARK BROWN	50	PUNE
160303	PART_P3	RED	90	MUMBAI
160304	PART_P4	RED	60	PUNE
160305	PART_P5	NAVY BLUE	80	MUMBAI

SELECT * FROM SUPPLIER;

S_NUM	S_NAME	STATUS	CITY
150304	JAMES WATSON	COMPLETED	NASHIK
150301	JOHN LINCOLN	COMPLETE	PUNE
150302	ROOT WILLIAMS	PENDING	PUNE
150303	WOOD THOMAS	STARTED	MUMBAI

SELECT * FROM SHIPMENT;

S_NUM	P_NUM	QUANTITY
150301	160301	40
150301	160302	40
150301	160303	40
150302	160301	60
150302	160302	40
150303	160302	40
150304	160301	60
150302	160303	100
150303	160301	50

SELECT SUPPLIER.S_NUM, SUPPLIER.S_NAME, PART.P_NUM, PART.P_NAME, SHIPMENT.QUANTITY FROM (SUPPLIER RIGHT JOIN SHIPMENT ON SUPPLIER.S_NUM=SHIPMENT.S_NUM) LEFT JOIN PART ON PART.P_NUM = SHIPMENT.P_NUM;

S_NUM	S_NAME	P_NUM	P_NAME	QUANTITY
150303	WOOD THOMAS	160301	PART_P1	50
150302	ROOT WILLIAMS	160301	PART_P1	60
150301	JOHN LINCOLN	160301	PART_P1	40
150304	JAMES WATSON	160301	PART_P1	60
150303	WOOD THOMAS	160302	PART_P2	40
150302	ROOT WILLIAMS	160302	PART_P2	40
150301	JOHN LINCOLN	160302	PART_P2	40
150302	ROOT WILLIAMS	160303	PART_P3	100
150301	JOHN LINCOLN	160303	PART_P3	40

SELECT SUPPLIER.S_NUM, SUPPLIER.S_NAME, PART.P_NUM, PART.P_NAME, SHIPMENT.QUANTITY FROM (SUPPLIER RIGHT JOIN SHIPMENT ON SUPPLIER.S_NUM=SHIPMENT.S_NUM) LEFT JOIN PART ON PART.P_NUM = SHIPMENT.P_NUM where SHIPMENT.QUANTITY > (SELECT AVG(QUANTITY) FROM SHIPMENT);

S_NUM	S_NAME	P_NUM	P_NAME	QUANTITY
150302	ROOT WILLIAMS	160301	PART_P1	60
150304	JAMES WATSON	160301	PART_P1	60
150302	ROOT WILLIAMS	160303	PART_P3	100

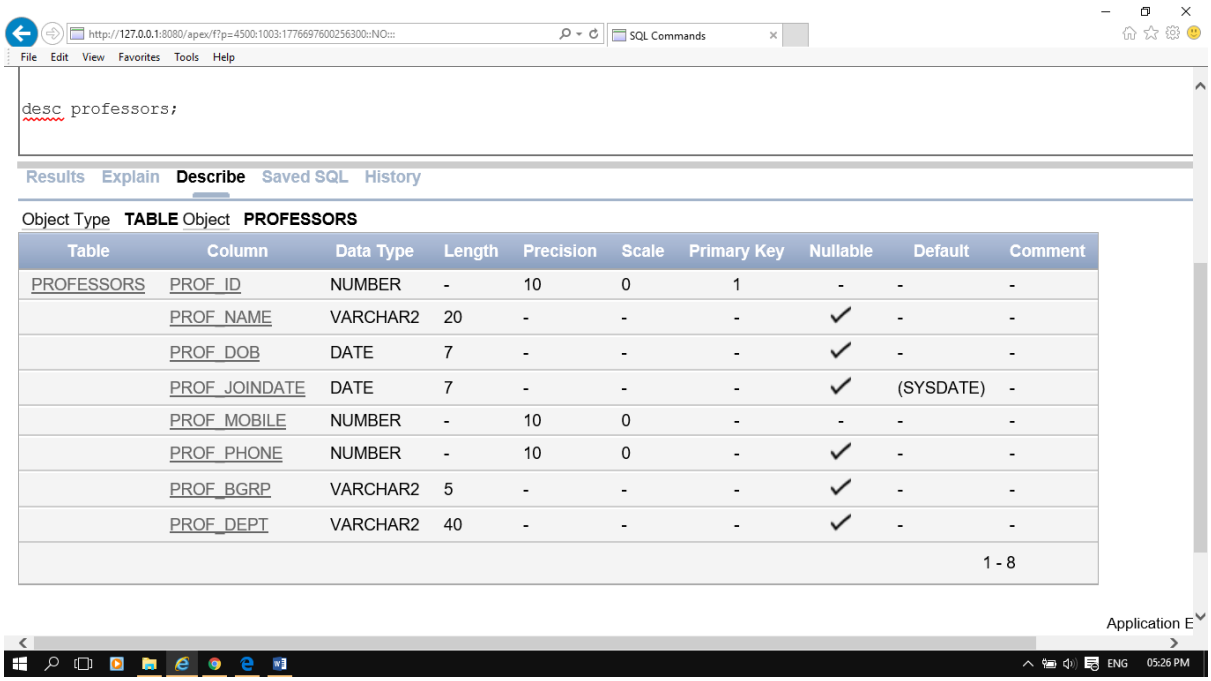
EXPERIMENT 2

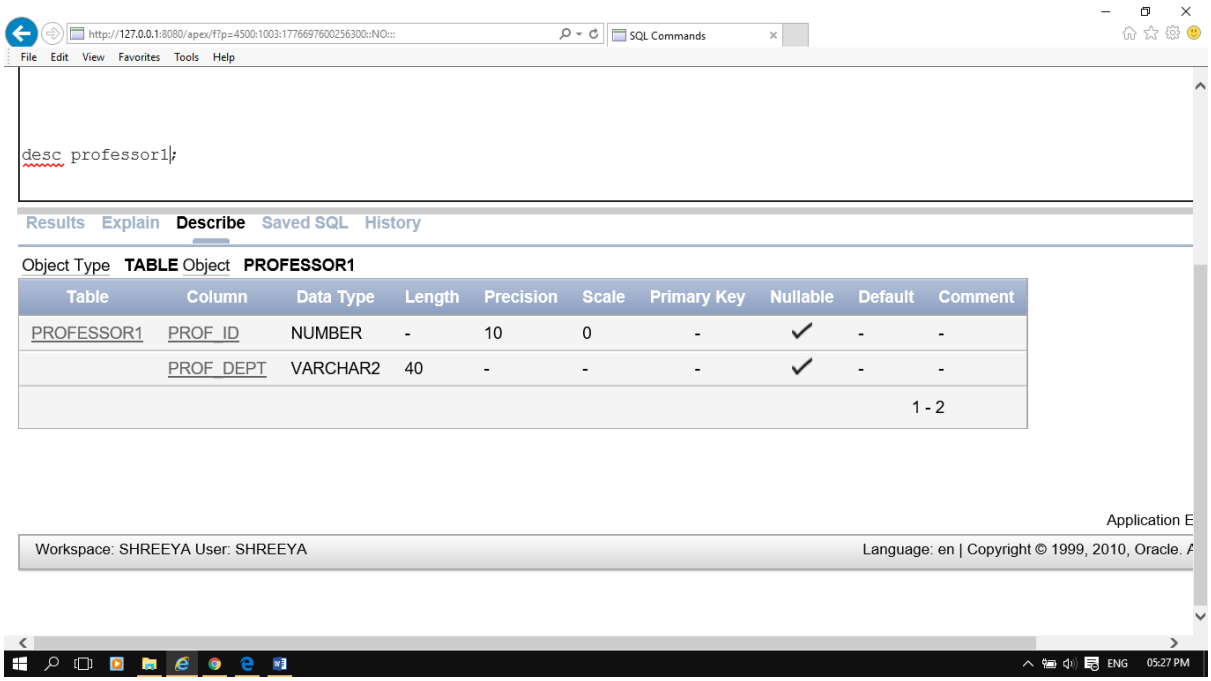
Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence, Synonym

CREATE TABLE:

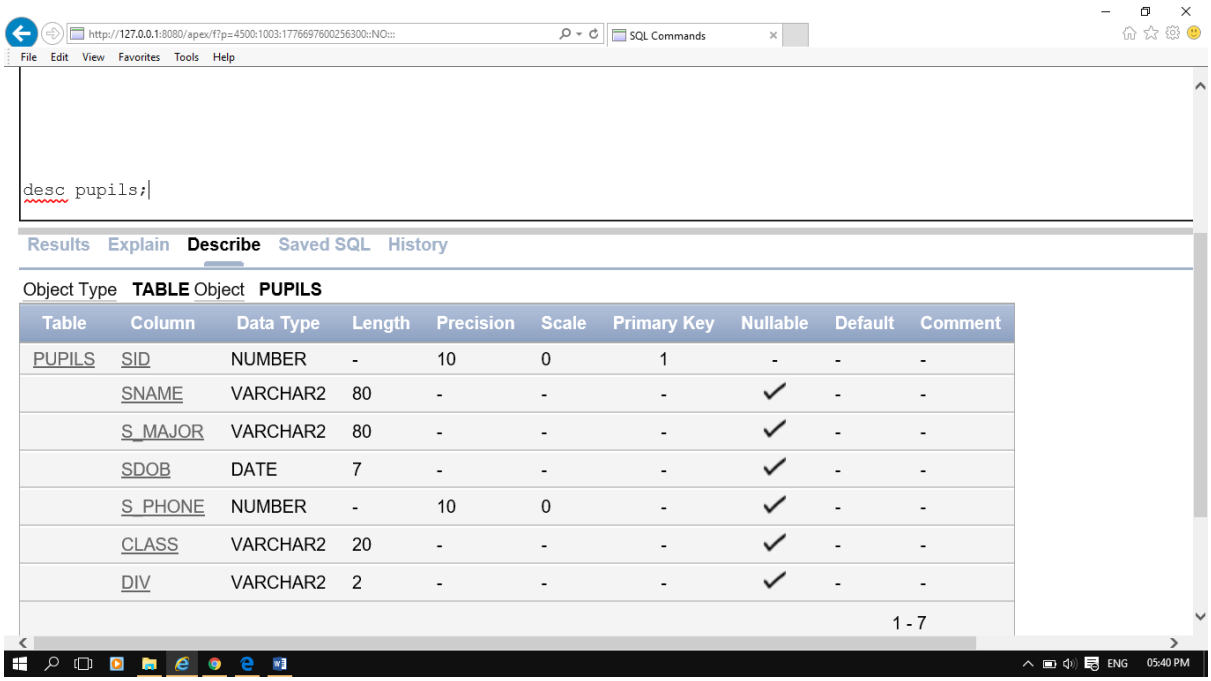
create table professors(prof_id number(10) primary key, prof_name varchar2(20), prof_dob date, prof_joindate date default(SYSDATE), prof_mobile number(10) NOT NULL, prof_phone number(10), prof_bgrp varchar2(5), prof_dept varchar2(40));

create table professor1 as select prof_id, prof_dept from professors;

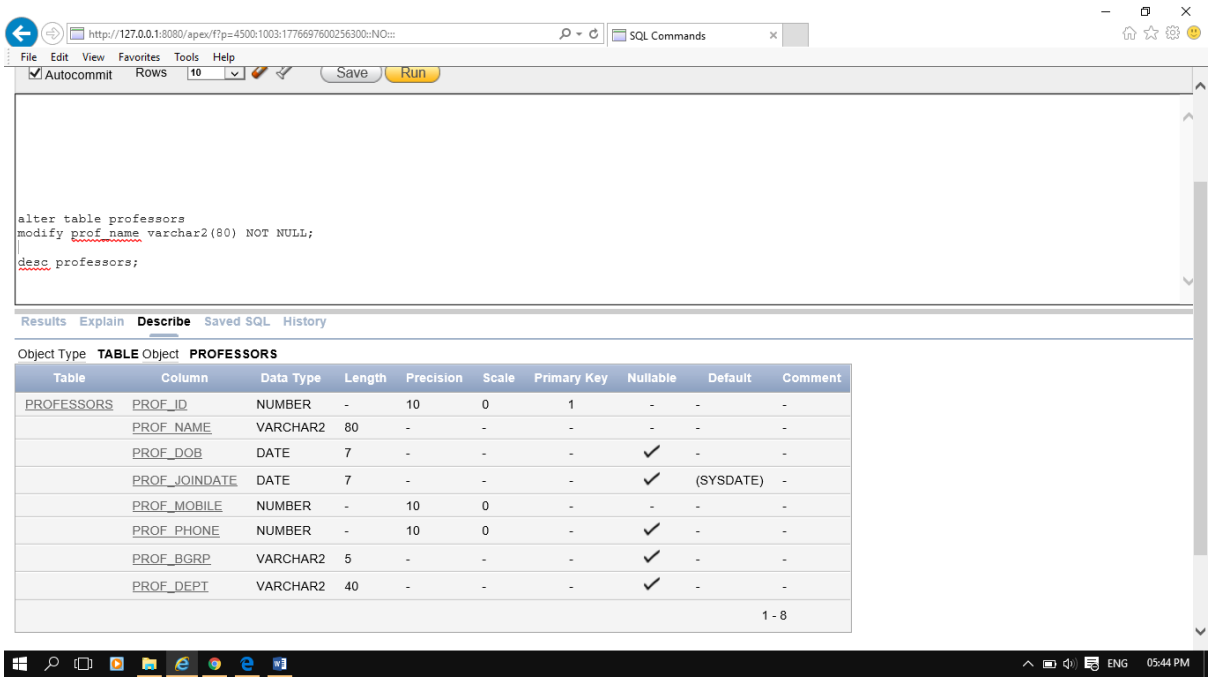


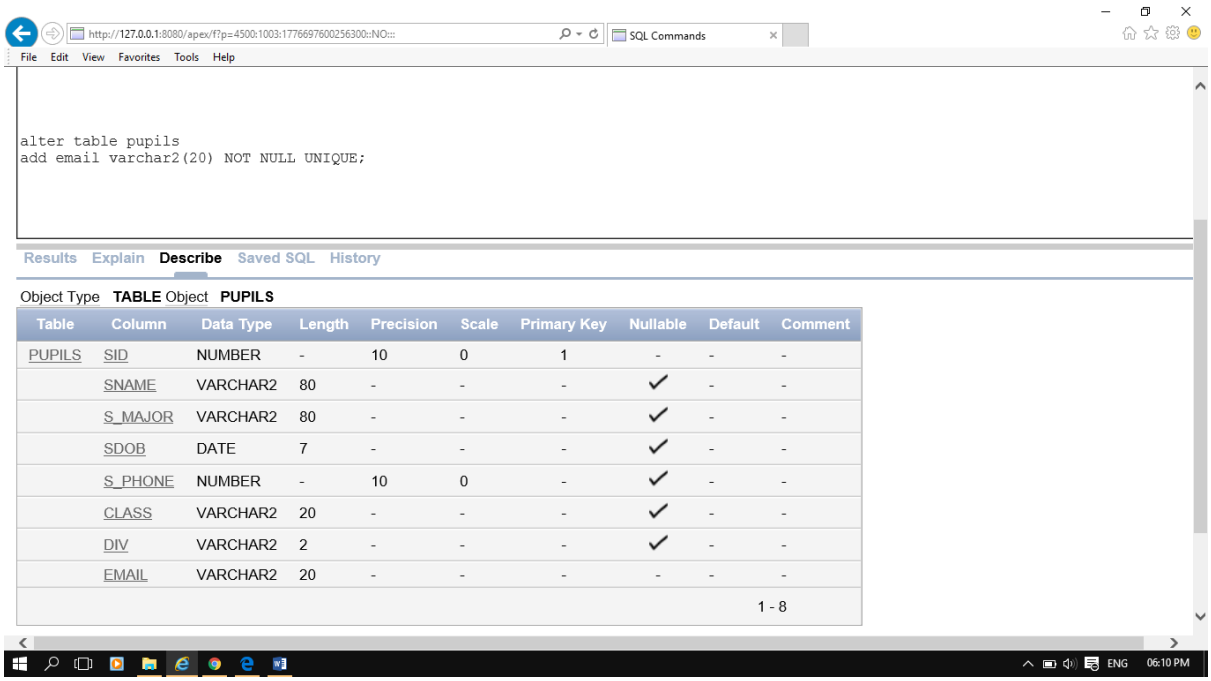


create table pupils(sid number(10), sname varchar2(80), s_major varchar2(80), sdob date, s_phone number(10), class varchar2(20), div varchar2(2), constraint pk111 primary key(sid));



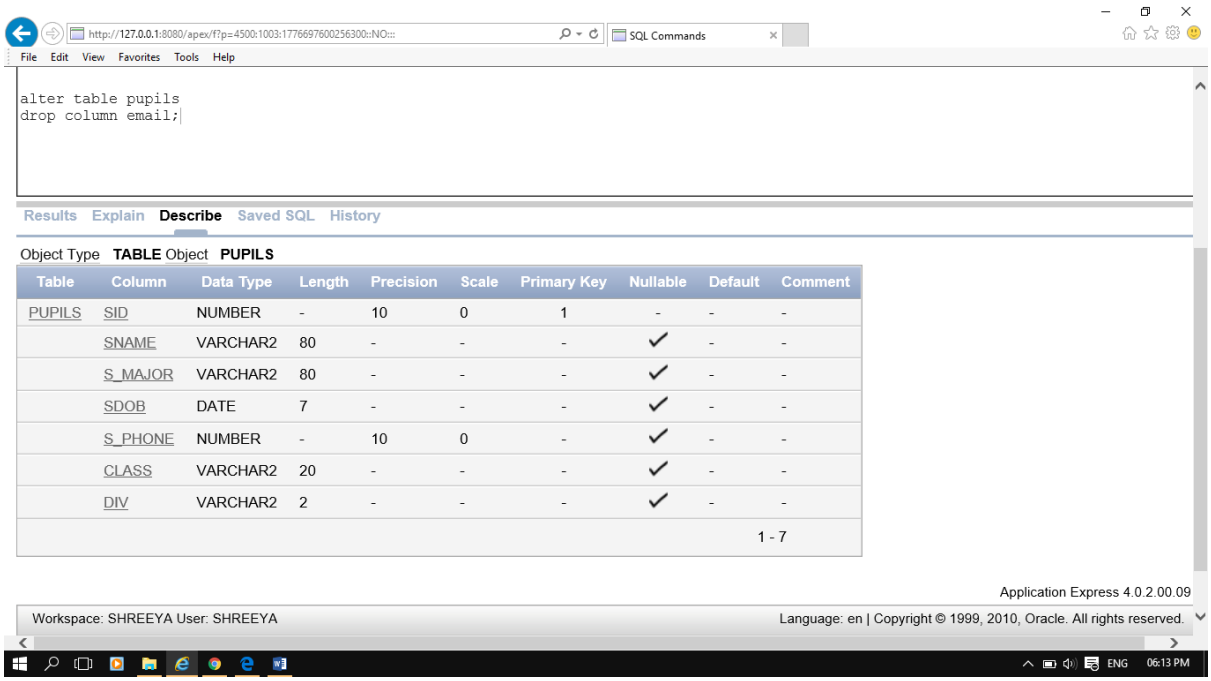

```
ALTER TABLE table_name
MODIFY column_name datatype;
```






```
ALTER TABLE table_name
ADD column_name datatype;
```

```
ALTER TABLE table_name
DROP COLUMN column_name;
```



Now we want to change the data type of the column named "DateOfBirth" in the "Persons" table.

We use the following SQL statement:

```
ALTER TABLE Persons
modify DateOfBirth varhcar2(7);
```

```
RENAME
alter table pupils
rename to pupil;
```

```
insert into pupil
with names as
(
SELECT 2, 'A', 'CO', '12-10-1999', 9900990099, 'TE', 'B' from DUAL
) SELECT * FROM names;
```

```
alter table pupil
rename to pupils
```

```
alter table pupils
drop constraint pk111
```

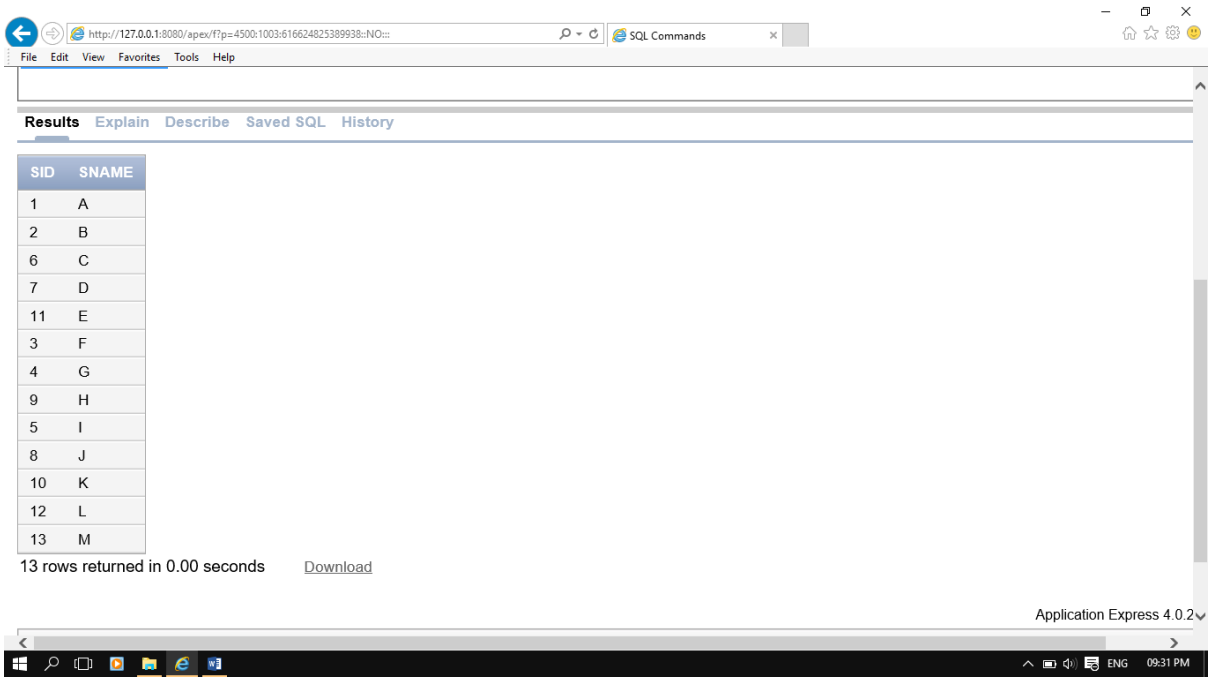
```
truncate table pupils
```

```
alter table pupils
add constraint pk111
primary key(sid)
```

```
insert into pupils values
(1, 'A', 'CO', '10-5-1999', 9763675555, 'TE', 'B')
```

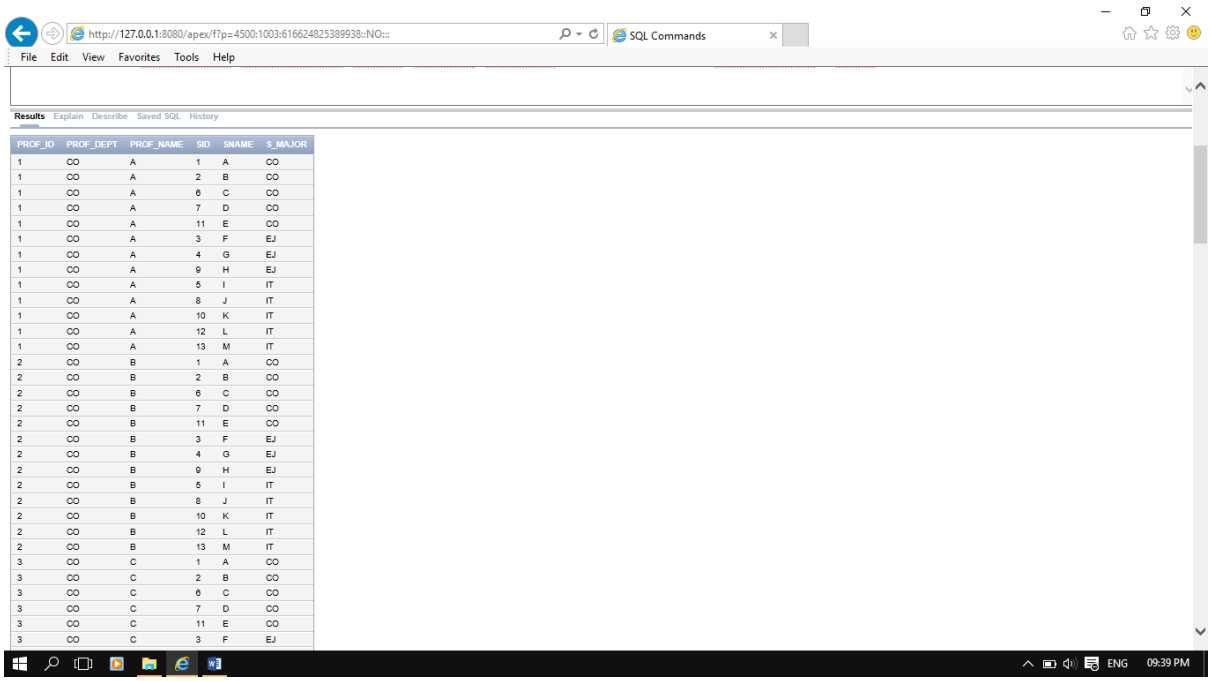
```
create view v1 as select sid, sname from pupils
```

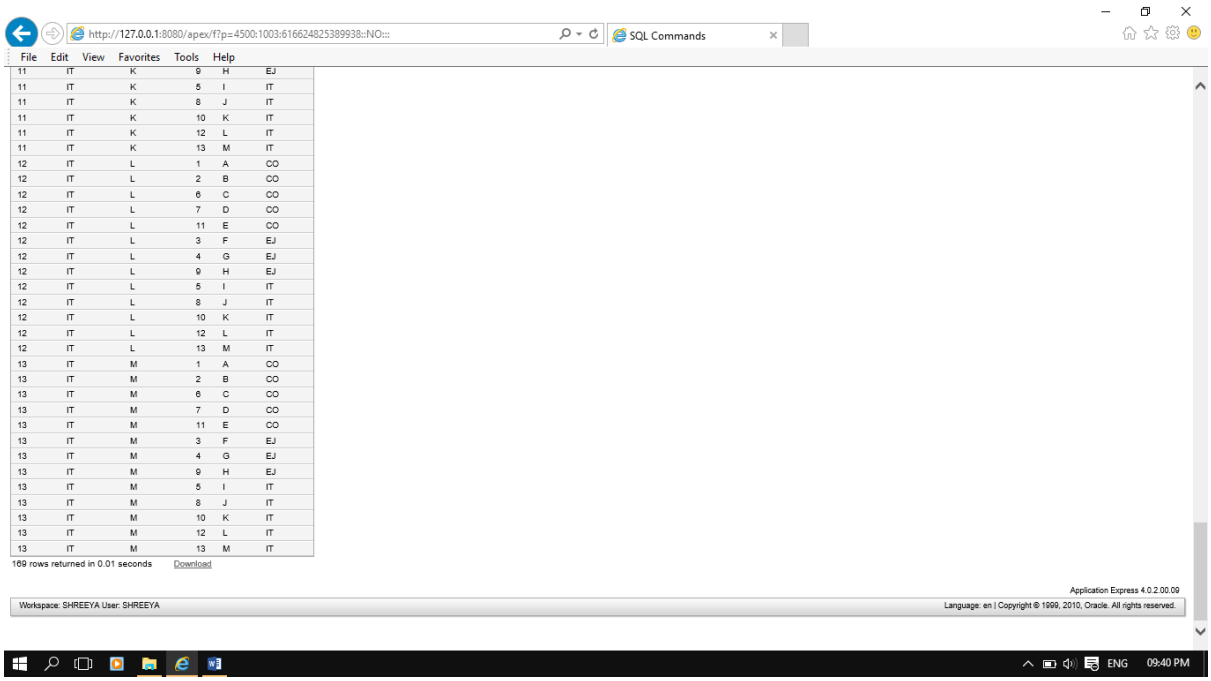
```
select * from v1
```



select professors.prof_id, professors.prof_dept, professors.prof_name, pupils.sid, pupils.sname, pupils.s_major from professors, pupils;

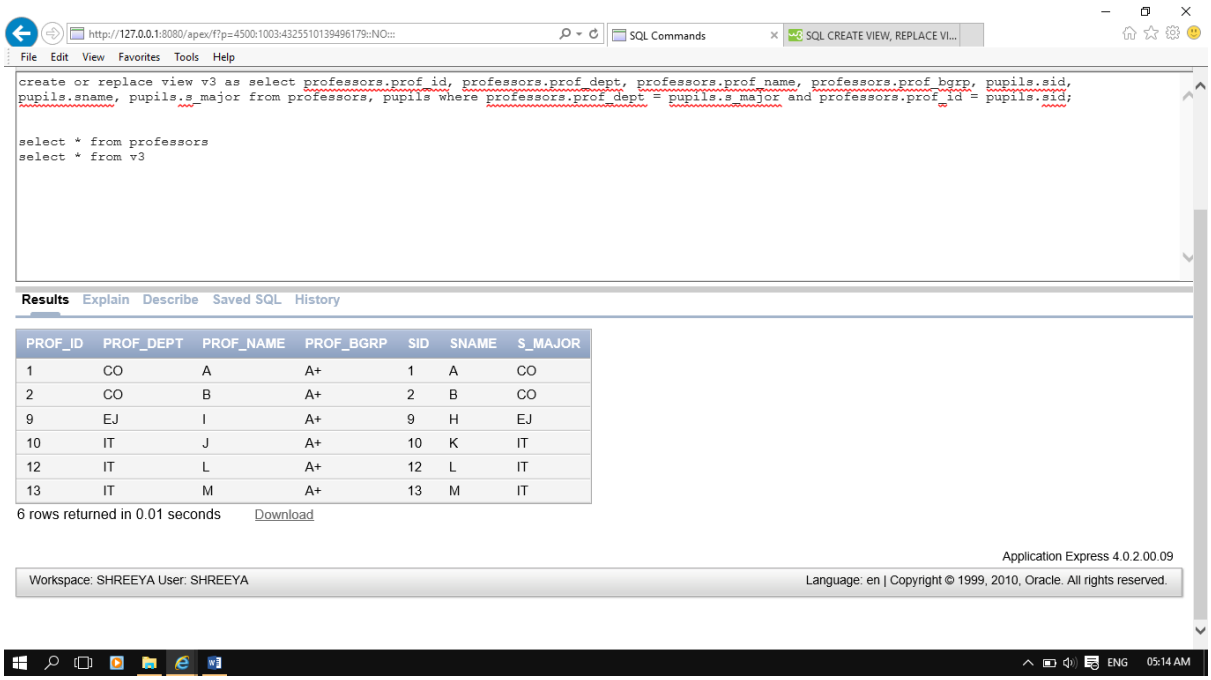
create view v2 as select professors.prof_id, professors.prof_dept, professors.prof_name, pupils.sid, pupils.sname, pupils.s_major from professors, pupils;





create or replace view v3 as select professors.prof_id, professors.prof_dept, professors.prof_name, professors.prof_bgrp, pupils.sid, pupils.sname, pupils.s_major from professors, pupils where professors.prof_dept = pupils.s_major and professors.prof_id = pupils.sid;

select * from professors
select * from v3



drop view v3;

➔ View dropped

create index id1 on professors(PROF_ID,

PROF_NAME,

PROF_DOB,

PROF_JOINDATE,

PROF_MOBILE,

PROF_PHONE,

PROF_BGRP,

PROF_DEPT);

drop index id1

```
CREATE SEQUENCE sequence_name
START WITH initial_value
INCREMENT BY increment_value
MINVALUE minimum value
MAXVALUE maximum value
CYCLE|NOCYCLE ;
```

sequence_name: Name of the sequence.

initial_value: starting value from where the sequence starts. Initial_value should be greater than or equal to minimum value and less than equal to maximum value.

increment_value: Value by which sequence will increment itself.

Increment_value can be positive or negative.

minimum_value: Minimum value of the sequence.

maximum_value: Maximum value of the sequence.

cycle: When sequence reaches its set_limit
it starts from beginning.

nocycle: An exception will be thrown
if sequence exceeds its max_value.

```
CREATE SEQUENCE sequence_1
start with 1
increment by 1
minvalue 0
maxvalue 100
cycle
```

```
create sequence seq1
start with 5
increment by 1
minvalue 5
maxvalue 66
CYCLE;
```

```
SELECT seq1.CURRVAL FROM DUAL;
```

```
alter sequence seq1
increment by -1
```

```
SELECT seq1.NEXTVAL FROM DUAL;
SELECT sequence_1.NEXTVAL FROM DUAL;
```

create synonym syn1 for professors

select * from syn1
create synonym syn2 for seq1

```
DECLARE
   <declarations section>
BEGIN
   <executable command(s)>
EXCEPTION
   <exception handling>
END;
```

DECLARE
msg varchar2(80) := 'HIII SHREEEYA';

BEGIN

dbms_output.put_line(msg);

End;

- ----------
- create table student(roll_no number(6) primary key, first_name varchar2(80) NOT NULL, middle_name varchar2(80), last_name varchar2(80) NOT NULL, phone_no number(10) NOT NULL, native_place varchar2(80) NOT NULL)

➔ Table created

- alter table student

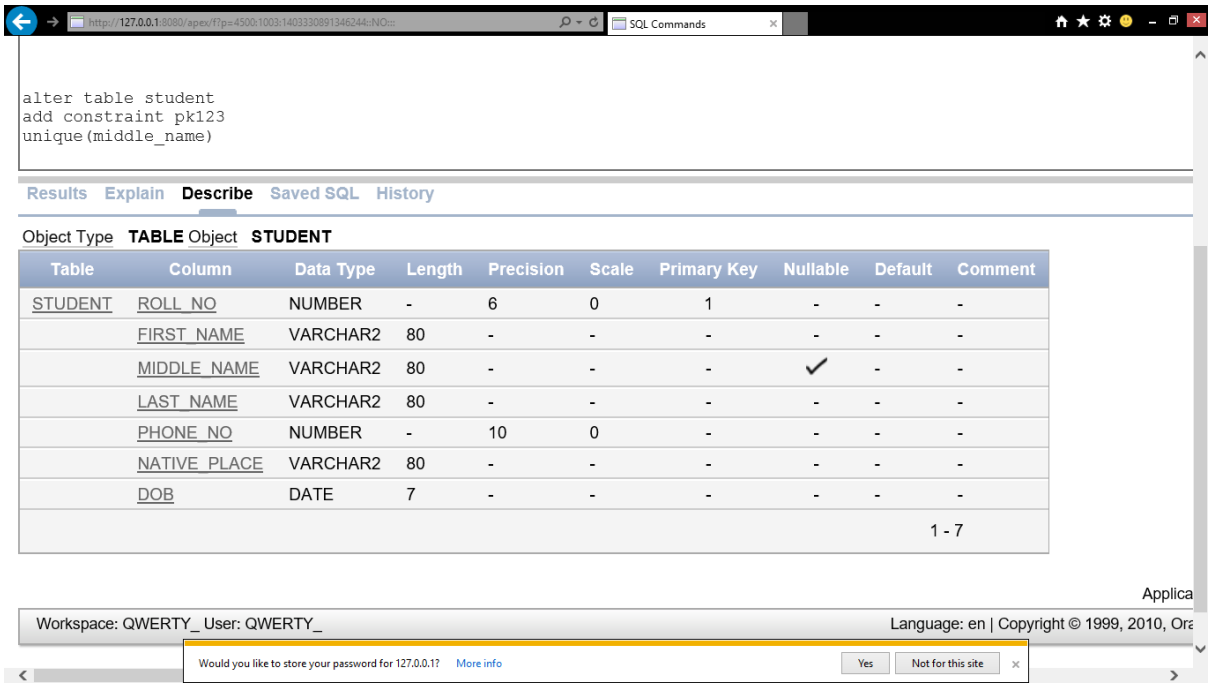
add constraint pk123
unique(middle_name)

➔ Table Altered

alter table student
MODIFY(dob date NOT NULL);

➔ Table altered

desc student



create table classes(room_no number(3), year_of_study char(4), div char(1), no_of_seats number(3))

→TABLE CREATED

alter table classes
add constraint pk1234 primary key(room_no)

alter table student
add room_no number(3)

alter table student
add constraint pk124 foreign key(room_no) references classes(room_no) on delete SET NULL;
(^^ If you execute again then error **ORA-02275: such a referential constraint already exists in the table**)

Datatypes in oracle
https://docs.oracle.com/cd/A58617_01/server.804/a58241/ch5.htm

alter table student
rename to pupils
→Table Altered

alter table pupils rename to student

alter table pupils rename to student

alter table student
rename column phone_no to mobile_no;

alter table student
rename column room_no to classroom_no;

alter table student
rename constraint
pk124 to pk1234

ERROR: **ORA-02264: name already used by an existing constraint**

(Pk1234 is already used in classes)

alter table student
rename constraint
pk124 to pk1

→Table altered

alter table classes
rename column
room_no to class_no

-->Table altered

create table instructor(inst_id number(6), inst_first_name varchar2(80), inst_middle_name varchar2(80), inst_last_name varchar2(80), year_of_joining number(4));

➔ Table Created

alter table instructor add constraint
pk2 primary key(inst_id)

➔ table altered

truncate table student

drop table student

VIEWS

In Oracle, view is a virtual table that does not physically exist. It is stored in Oracle data dictionary and do not store any data. It can be executed when called.

A view is created by a query joining one or more tables.

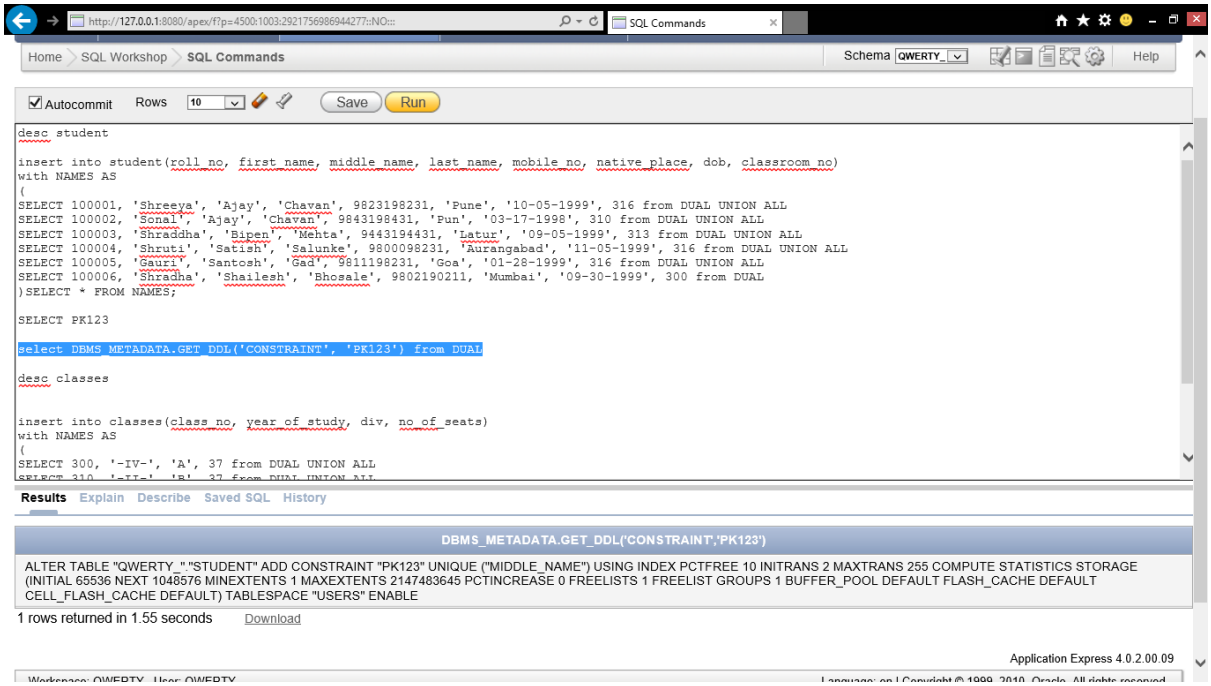
```
insert into student(roll_no, first_name, middle_name, last_name, mobile_no, native_place, dob, classroom_no)
with NAMES AS
(
SELECT 100001, 'Shreeya', 'Ajay', 'Chavan', 9823198231, 'Pune', '10-05-1999', 316 from DUAL UNION ALL
SELECT 100002, 'Sonal', 'Ajay', 'Chavan', 9843198431, 'Pune', '03-17-1998', 310 from DUAL UNION ALL
SELECT 100003, 'Shraddha', 'Bipen', 'Mehta', 9443194431, 'Latur', '09-05-1999', 313 from DUAL UNION ALL
SELECT 100004, 'Shruti', 'Satish', 'Salunke', 9800098231, 'Aurangabad', '11-05-1999', 316 from DUAL UNION ALL
SELECT 100005, 'Gauri', 'Santosh', 'Gad', 9811198231, 'Goa', '01-28-1999', 316 from DUAL UNION ALL
SELECT 100006, 'Shraddha', 'Shailesh', 'Bhosale', 9802190211, 'Mumbai', '09-30-1999', 300 from DUAL UNION ALL
SELECT 100006, 'Shraddha', 'Shailesh', 'Bhosale', 9802190211, 'Mumbai', '09-30-1999', 300 from DUAL
)SELECT * FROM NAMES;
```

ERROR: **ORA-02291: integrity constraint (QWERTY_.PK1) violated - parent key not found**

```
insert into classes(class_no, year_of_study, div, no_of_seats)
with NAMES AS
(
SELECT 300, '-IV-', 'A', 37 from DUAL UNION ALL
SELECT 310, '-II-', 'B', 37 from DUAL UNION ALL
SELECT 313, '-I--', 'A', 37 from DUAL UNION ALL
SELECT 316, '-III', 'B', 34 from DUAL UNION ALL
SELECT 315, '-II-', 'A', 32 from DUAL
)SELECT * FROM NAMES;
```

5 rows inserted

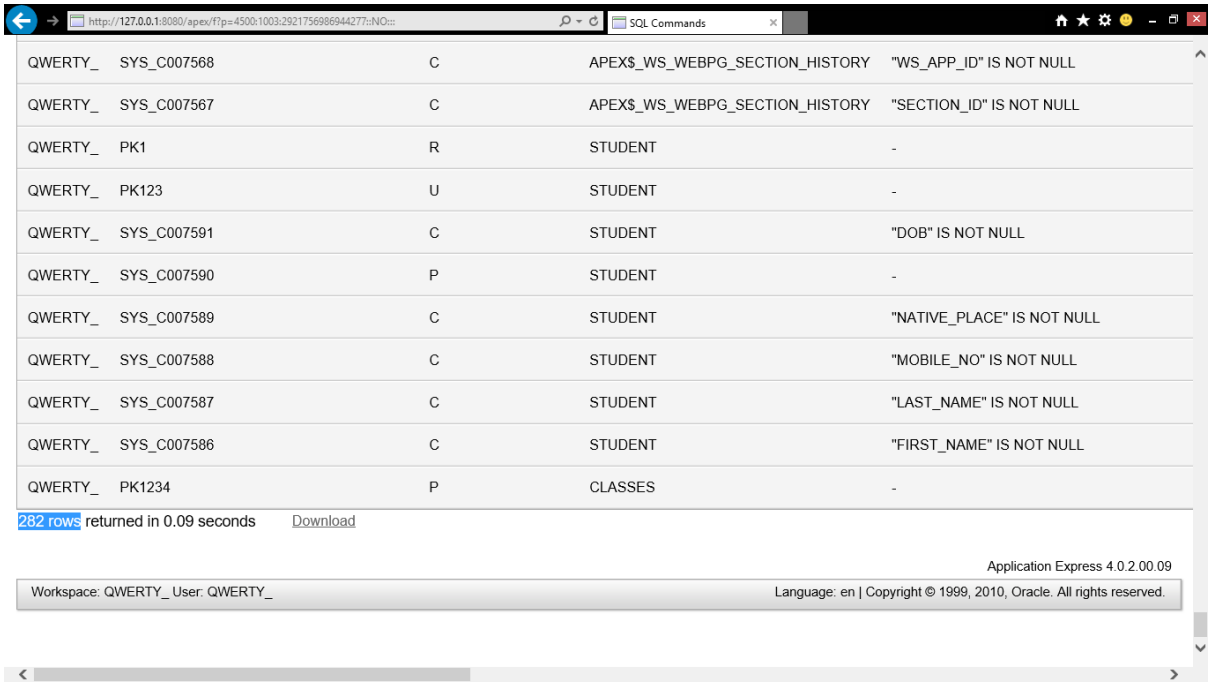
```
select DBMS_METADATA.GET_DDL('CONSTRAINT', 'PK123') from DUAL
```



insert into student(roll_no, first_name, middle_name, last_name, mobile_no, native_place, dob, classroom_no)
with NAMES AS
(
SELECT 100001, 'Shreeya', 'Ajay', 'Chavan', 9823198231, 'Pune', '10-05-1999', 316 from DUAL UNION ALL
SELECT 100002, 'Sonal', 'A.', 'Chavan', 9843198431, 'Pune', '03-17-1998', 310 from DUAL UNION ALL
SELECT 100003, 'Shraddha', 'Bipen', 'Mehta', 9443194431, 'Latur', '09-05-1999', 313 from DUAL UNION ALL
SELECT 100004, 'Shruti', 'Satish', 'Salunke', 9800098231, 'Aurangabad', '11-05-1999', 316 from DUAL UNION ALL
SELECT 100005, 'Gauri', 'Santosh', 'Gad', 9811198231, 'Goa', '01-28-1999', 316 from DUAL UNION ALL
SELECT 100006, 'Shradha', 'Shailesh', 'Bhosale', 9802190211, 'Mumbai', '09-30-1999', 300 from DUAL
)SELECT * FROM NAMES;

6 Rows inserted

select * from ALL_CONSTRAINTS



QWERTY_	SYS_C007568	C	APEX$_WS_WEBPG_SECTION_HISTORY	"WS_APP_ID" IS NOT NULL
QWERTY_	SYS_C007567	C	APEX$_WS_WEBPG_SECTION_HISTORY	"SECTION_ID" IS NOT NULL
QWERTY_	PK1	R	STUDENT	-
QWERTY_	PK123	U	STUDENT	-
QWERTY_	SYS_C007591	C	STUDENT	"DOB" IS NOT NULL
QWERTY_	SYS_C007590	P	STUDENT	-
QWERTY_	SYS_C007589	C	STUDENT	"NATIVE_PLACE" IS NOT NULL
QWERTY_	SYS_C007588	C	STUDENT	"MOBILE_NO" IS NOT NULL
QWERTY_	SYS_C007587	C	STUDENT	"LAST_NAME" IS NOT NULL
QWERTY_	SYS_C007586	C	STUDENT	"FIRST_NAME" IS NOT NULL
QWERTY_	PK1234	P	CLASSES	-

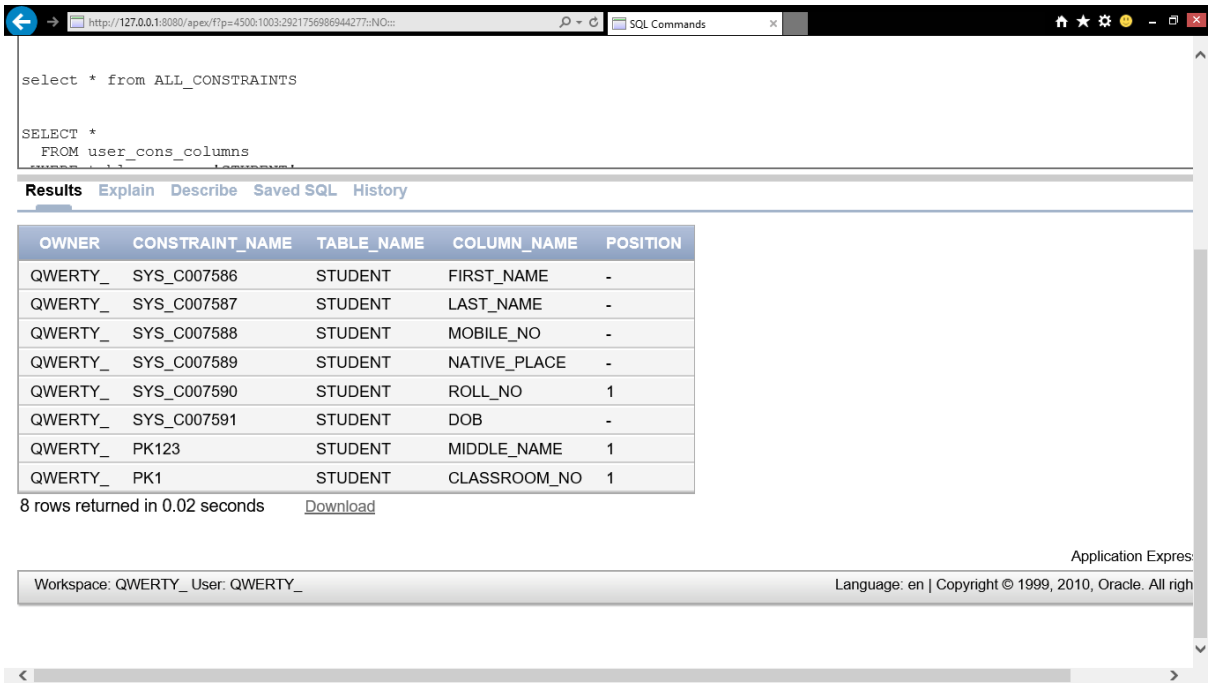
282 rows returned in 0.09 seconds Download

Application Express 4.0.2.00.09

Workspace: QWERTY_ User: QWERTY_ Language: en | Copyright © 1999, 2010, Oracle. All rights reserved.

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	SEARCH_CONDITION	R_OWNR

```
SELECT *
  FROM user_cons_columns
 WHERE table_name = 'STUDENT';
```



```
select * from ALL_CONSTRAINTS


SELECT *
  FROM user_cons_columns
```

Results Explain Describe Saved SQL History

OWNER	CONSTRAINT_NAME	TABLE_NAME	COLUMN_NAME	POSITION
QWERTY_	SYS_C007586	STUDENT	FIRST_NAME	-
QWERTY_	SYS_C007587	STUDENT	LAST_NAME	-
QWERTY_	SYS_C007588	STUDENT	MOBILE_NO	-
QWERTY_	SYS_C007589	STUDENT	NATIVE_PLACE	-
QWERTY_	SYS_C007590	STUDENT	ROLL_NO	1
QWERTY_	SYS_C007591	STUDENT	DOB	-
QWERTY_	PK123	STUDENT	MIDDLE_NAME	1
QWERTY_	PK1	STUDENT	CLASSROOM_NO	1

8 rows returned in 0.02 seconds Download

Application Expres

Workspace: QWERTY_ User: QWERTY_ Language: en | Copyright © 1999, 2010, Oracle. All righ

```
alter table instructor
add constraint pk3
primary key(inst_id)
```

Table altered

```
insert into instructor(inst_id, inst_first_name, inst_middle_name, inst_last_name, year_of_joining)
with NAMES AS
(
 SELECT 1, 'Ajay', 'V.', 'Chavan', 2014 from DUAL UNION ALL
 SELECT 2, 'Vijay', 'V.', 'Bhosale', 2014 from DUAL UNION ALL
 SELECT 3, 'Sujay', 'V.', 'Salunke', 2014 from DUAL UNION ALL
 SELECT 4, 'Dhananjay', 'V.', 'Paiyawal', 2014 from DUAL UNION ALL
 SELECT 5, 'MrityunAjay', 'V.', 'Panchbhai', 2014 from DUAL UNION ALL
 SELECT 6, 'Jay', 'V.', 'Ramdasi', 2014 from DUAL UNION ALL
 SELECT 7, 'Sanjay', 'V.', 'Iyer', 2014 from DUAL UNION ALL
 SELECT 8, 'Kiran', 'V.', 'Mehta', 2014 from DUAL UNION ALL
 SELECT 9, 'Naveen', 'S.', 'Reddy', 2014 from DUAL UNION ALL
 SELECT 10, 'Sanchit', 'R.', 'Jain', 2014 from DUAL UNION ALL
 SELECT 11, 'Corey', 'M.', 'Schafer', 2014 from DUAL
)
SELECT * FROM NAMES
```

11 row(s) inserted.

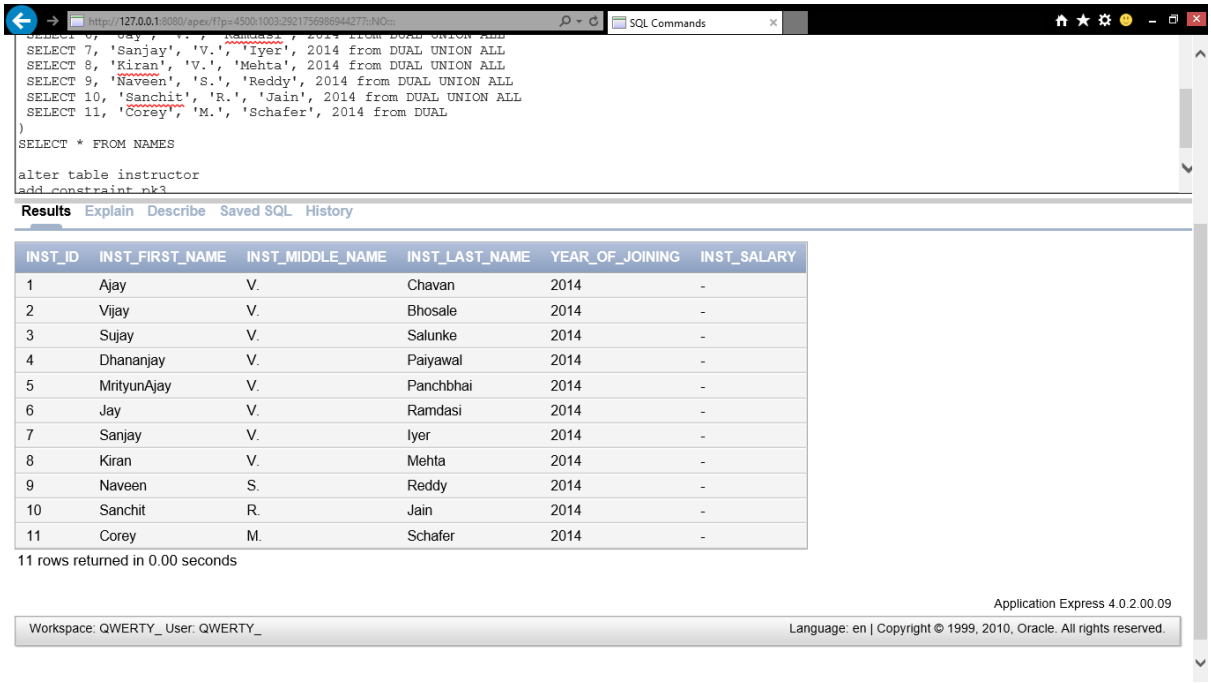
```
alter table instructor
add inst_salary number(10);
```

➔ Table altered

```
alter table instructor
add inst_sal_incre number(10) NOT NULL;
```

->ORA-01758: table must be empty to add mandatory (NOT NULL) column

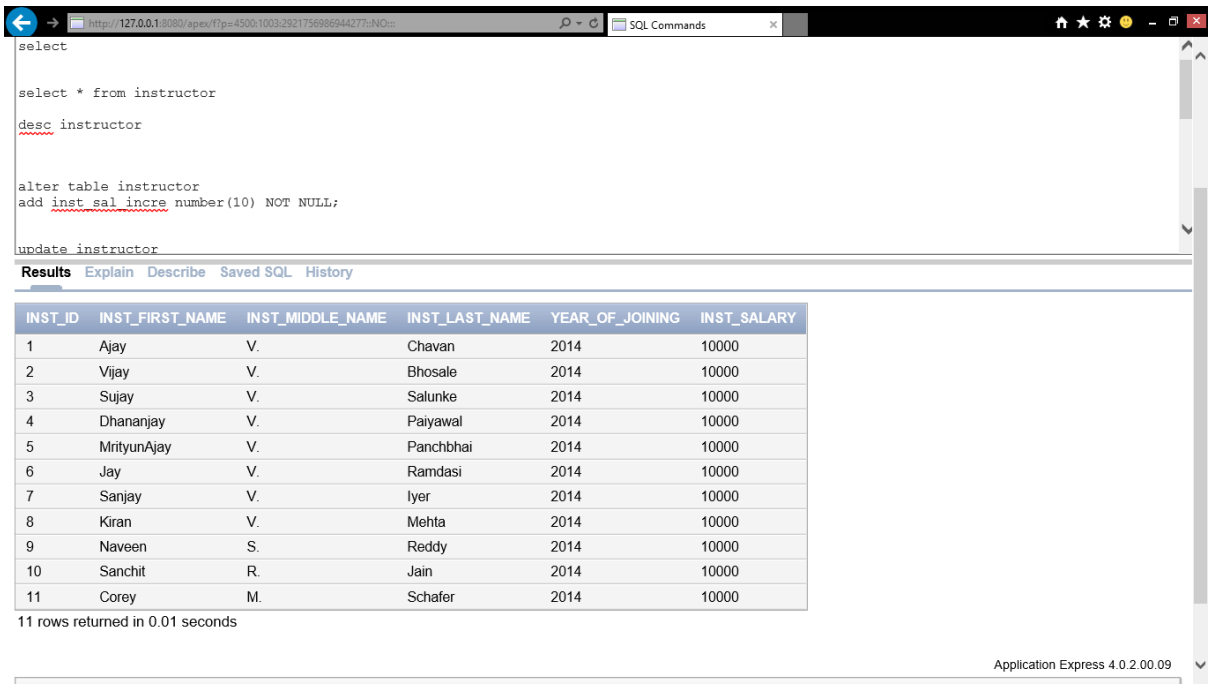
```
Select * from instructor
```



update instructor
set inst_salary = 10000
where inst_salary IS NULL;

➔ 11 row(s) updated

select * from instructor



create table stud_inst(sid number(6), constraint fp1 foreign key(sid) references student(roll_no))

➔ Table created

alter table stud_inst
add constraint fk2 foreign key(inst_id) references instructor(inst_id)

---

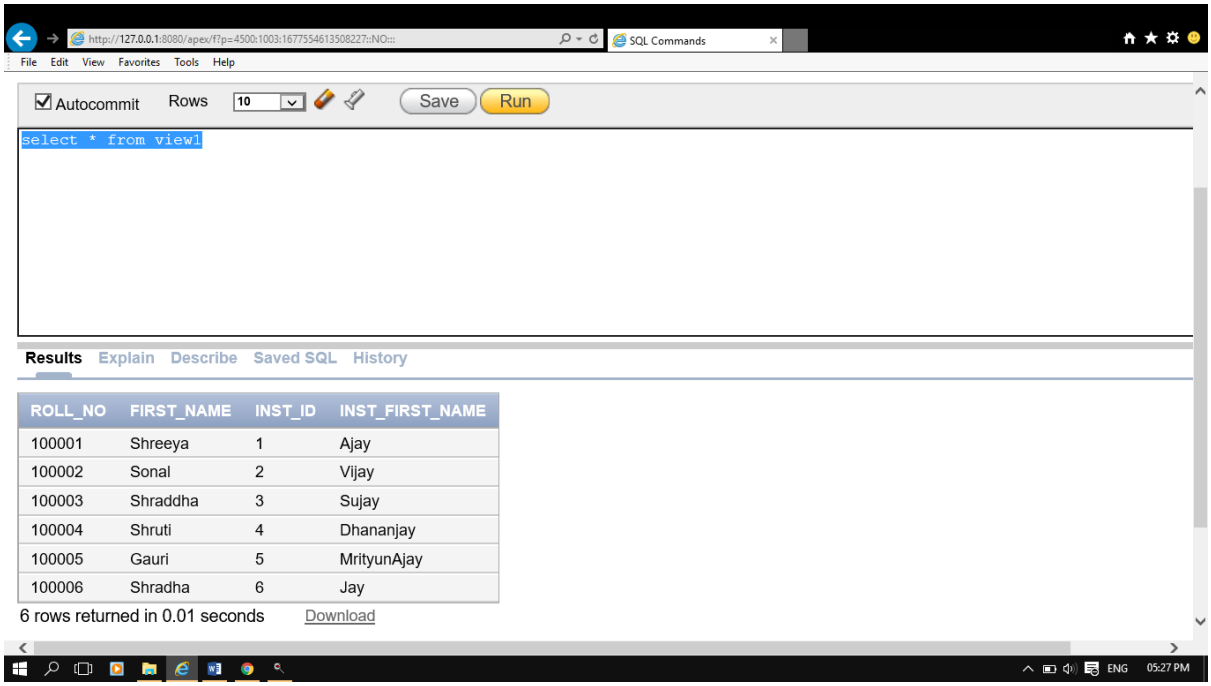
create view view1 as
(select student.roll_no, student.first_name, instructor.inst_id, instructor.inst_first_name from student, instructor, stud_inst
where student.roll_no = stud_inst.sid and instructor.inst_id = stud_inst.inst_id);

➔ View created

create view view2 as
(select student.roll_no, student.first_name, instructor.inst_id, instructor.inst_first_name from student, instructor, stud_inst
where student.roll_no = stud_inst.sid and instructor.inst_id = stud_inst.inst_id);

➔ View created

select * from view1



```
create or replace view view2 as
(select student.roll_no, student.first_name, student.last_name, instructor.inst_id,
instructor.inst_first_name from student, instructor, stud_inst
where student.roll_no = stud_inst.sid and instructor.inst_id = stud_inst.inst_id);

->view created


create or replace view view2 as
(select student.roll_no, student.first_name, student.last_name, student.middle_name,
instructor.inst_id, instructor.inst_first_name from student, instructor, stud_inst
where student.roll_no = stud_inst.sid and instructor.inst_id = stud_inst.inst_id);

->
View created

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

rename view2 to view22


->Statement Processed
----------------------------------
```

Question: Can you update the data in an Oracle VIEW?

Answer: A VIEW in Oracle is created by joining one or more tables. When you update record(s) in a VIEW, it updates the records in the underlying tables that make up the View.

So, yes, you can update the data in an Oracle VIEW providing you have the proper privileges to the underlying Oracle tables.

Question: Does the Oracle View exist if the table is dropped from the database?

Answer: Yes, in Oracle, the VIEW continues to exist even after one of the tables (that the Oracle VIEW is based on) is dropped from the database. However, if you try to query the Oracle VIEW after the table has been dropped, you will receive a message indicating that the Oracle VIEW has errors.

If you recreate the table (the table that you had dropped), the Oracle VIEW will again be fine

```
-------------------------------------
rename view22 to view2
    ➔ Statement Processed
-------------------------------------


create view view3
as
```

select class_no, year_of_study, div from classes;

➔ View Created

---

create table classes2 as select * from classes

➔ Table created

CREATE VIEW view4 as select class_no, year_of_study, div from classes2

➔ View created

---

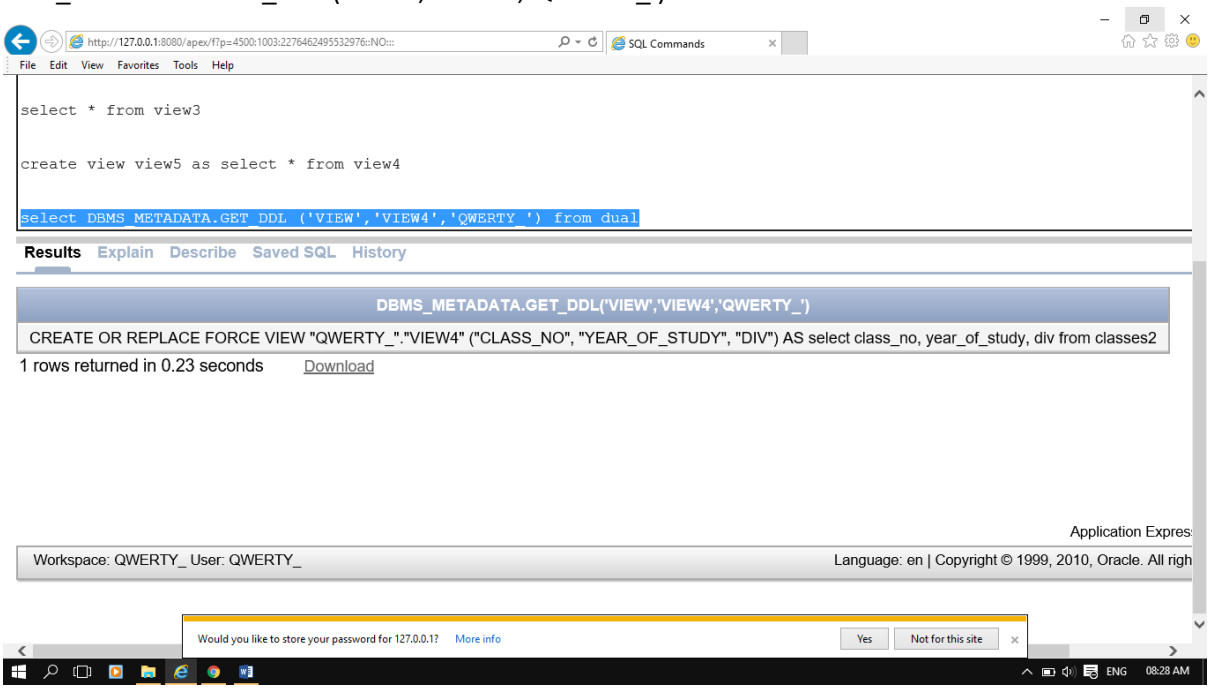
create view view5 as select * from view4

➔ View created

---

select DBMS_METADATA.GET_DDL ('VIEW','VIEW4','QWERTY_') from dual

➔



---

select * from ALL_VIEWS

->(ALL VIEWS ARE DISPLAYED)

You can describe views using

- Desc view-name
- ---

---

select DBMS_METADATA.GET_DDL ('VIEW','VIEW4','QWERTY_') from dual
---

create view classes as select class_no, year_of_study, div from classes2

➔ Name is already used by an existing object

An updatable view is one you can use to insert, update, or delete base table rows. You can create a view to be inherently updatable, or you can create an INSTEAD OF trigger on any view to make it updatable.

To learn whether and in what ways the columns of an inherently updatable view can be modified, query the USER_UPDATABLE_COLUMNS data dictionary view. The information displayed by this view is meaningful only for inherently updatable views. For a view to be inherently updatable, the following conditions must be met:

- Each column in the view must map to a column of a single table. For example, if a view column maps to the output of a TABLE clause (an unnested collection), then the view is not inherently updatable.
- The view must not contain any of the following constructs:
  - A set operator
  - a DISTINCT operator
  - An aggregate or analytic function
  - A GROUP BY, ORDER BY, MODEL, CONNECT BY, or START WITH clause
  - A collection expression in a SELECT list
  - A subquery in a SELECT list
  - A subquery designated WITH READ ONLY
  - Joins, with some exceptions, as documented in Oracle Database Administrator's Guide

In addition, if an inherently updatable view contains pseudocolumns or expressions, then you cannot update base table rows with an UPDATE statement that refers to any of these pseudocolumns or expressions.

If you want a join view to be updatable, then all of the following conditions must be true:

- The DML statement must affect only one table underlying the join.
- For an INSERT statement, the view must not be created WITH CHECK OPTION, and all columns into which values are inserted must come from a key-preserved table. A key-preserved table is one for which every primary key or unique key value in the base table is also unique in the join view.
- For an UPDATE statement, all columns updated must be extracted from a key-preserved table. If the view was created WITH CHECK OPTION, then join columns and columns taken from tables that are referenced more than once in the view must be shielded from UPDATE.
- For a DELETE statement, if the join results in more than one key-preserved table, then Oracle Database deletes from the first table named in the FROM clause, whether or not the view was created WITH CHECK OPTION

INDEX

https://stackoverflow.com/questions/1652995/in-oracle-is-it-possible-to-insert-or-update-a-record-through-a-view
---

An index is used to speed up searching in the database. MySQL have some good documentation on the subject (which is relevant for other SQL servers as well): http://dev.mysql.com/doc/refman/5.0/en/mysql-indexes.html
An index can be used to efficiently find all rows matching some column in your query and then walk through only that subset of the table to find exact matches. If you don't have indexes on any column in the `WHERE` clause, the `SQL` server has to walk through *the whole table* and check every row to see if it matches, which may be a slow operation on big tables.
The index can also be a `UNIQUE` index, which means that you cannot have duplicate values in that column, or a `PRIMARY KEY` which in some storage engines defines where in the database file the value is stored.
In MySQL you can use `EXPLAIN` in front of your `SELECT` statement to see if your query will make use of any index. This is a good start for troubleshooting performance problems. Read more here: http://dev.mysql.com/doc/refman/5.0/en/explain.html
---

https://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_5010.htm#i2084975
index
---

create index idx1 on student(roll_no, first_name)

➔ Index created

---

drop index idx1

➔ Index dropped

Some reasons for dropping an index include:

- The index is no longer required.

- The index is not providing anticipated performance improvements for queries issued against the associated table. For example, the table might be very small, or there might be many rows in the table but very few index entries.
- Applications do not use the index to query the data.
- The index has become invalid and must be dropped before being rebuilt.
- The index has become too fragmented and must be dropped before being rebuilt.

When you drop an index, all extents of the index segment are returned to the containing tablespace and become available for other objects in the tablespace.

How you drop an index depends on whether you created the index explicitly with a `CREATE INDEX` statement, or implicitly by defining a key constraint on a table. If you created the index explicitly with the `CREATE INDEX` statement, then you can drop the index with the `DROP INDEX` statement. The following statement drops the `emp_ename` index:

```
DROP INDEX emp_ename;
```

You cannot drop only the index associated with an enabled `UNIQUE` key or `PRIMARY KEY` constraint. To drop a constraints associated index, you must disable or drop the constraint itself.

https://docs.oracle.com/cd/B28359_01/server.111/b28310/indexes006.htm#ADMIN11737

---

https://docs.oracle.com/cd/B28359_01/server.111/b28310/indexes005.htm#ADMIN11736

managing indexes

SEQUENCE

https://www.techonthenet.com/oracle/sequences.php#targetText=A%20sequence%20is%20an%20object,act%20as%20a%20primary%20key.

---

create synonym syn1 for student

➔ Synonym created’

---

drop table syn1

➔ **ORA-00942: table or view does not exist**

➔

0.00 seconds

Oerr utility

http://www.dba-oracle.com/t_oerr.htm

EXPERIMENT 2

Design at least 10 SQL queries for suitable database application using SQL DML statements: Insert, Select, Update, Delete with operators, functions, and set operator

Insert:
insert into student values(100007, 'Patricia', 'Patrick', 'D''gama', 9823982311, 'Tiruvananthpuram, Kerala', '01-10-1977', 313)

➔ 1 row(s) inserted

---

insert into student(roll_no, first_name, middle_name, last_name, mobile_no, native_place, dob, classroom_no)

with NAMES as

(

SELECT 100008, 'Amruta', 'K','Raut',9823982311, 'Tiruvananthpuram, Kerala','01-10-1999',313 FROM DUAL UNION ALL

SELECT 100009, 'Namrta', 'C','Pandey',9821111111,'Tiruvanantpuram, ''eral'''','01-10-1999',313 FROM DUAL UNION ALL

SELECT 100010, 'Kirti', 'R','Maheshwr',9823982311, 'Tiruvananthpuram, Kerala','01-10-1999',313 FROM DUAL

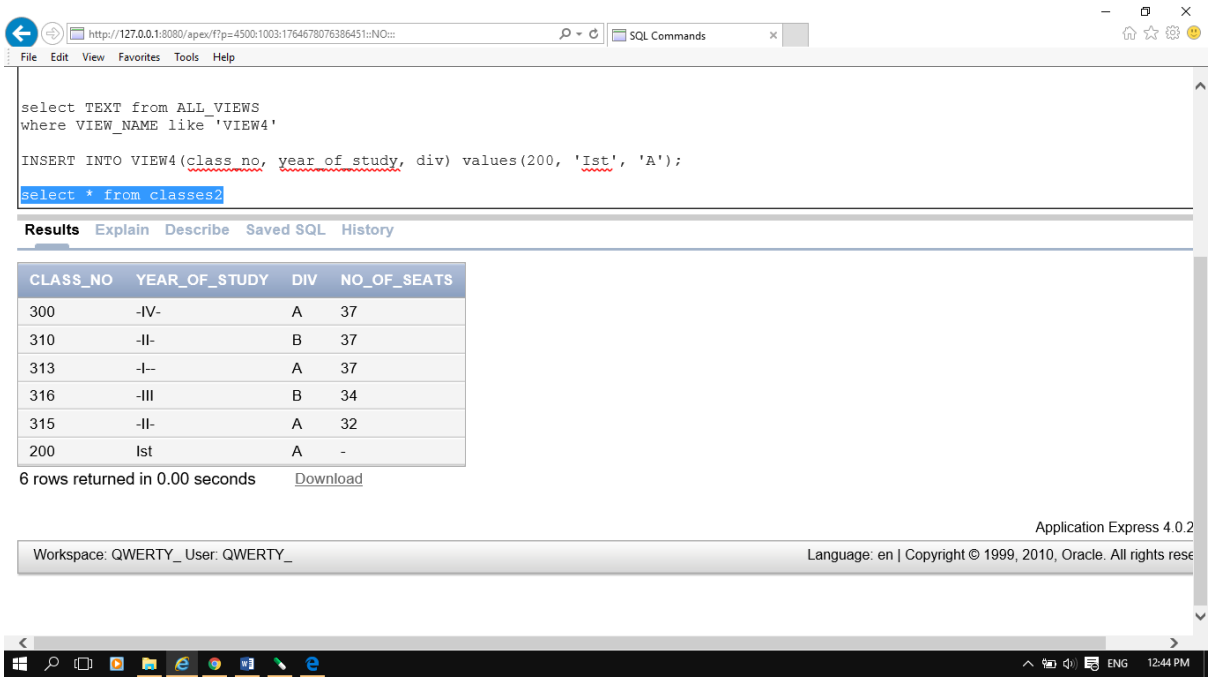
)select * from NAMES

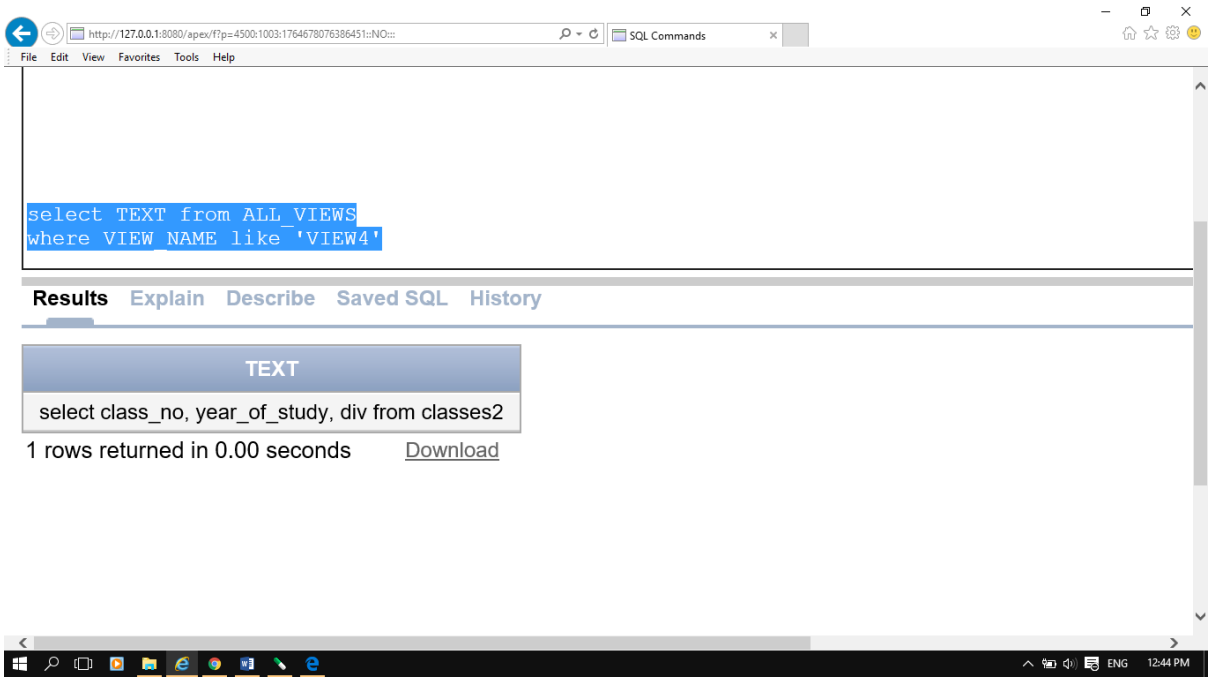
➔ 3 row(s) inserted

INSERT INTO VIEW4(class_no, year_of_study, div) values(200, 'Ist', 'A');

1 row(s) inserted

---
select * from classes2





delete from view4

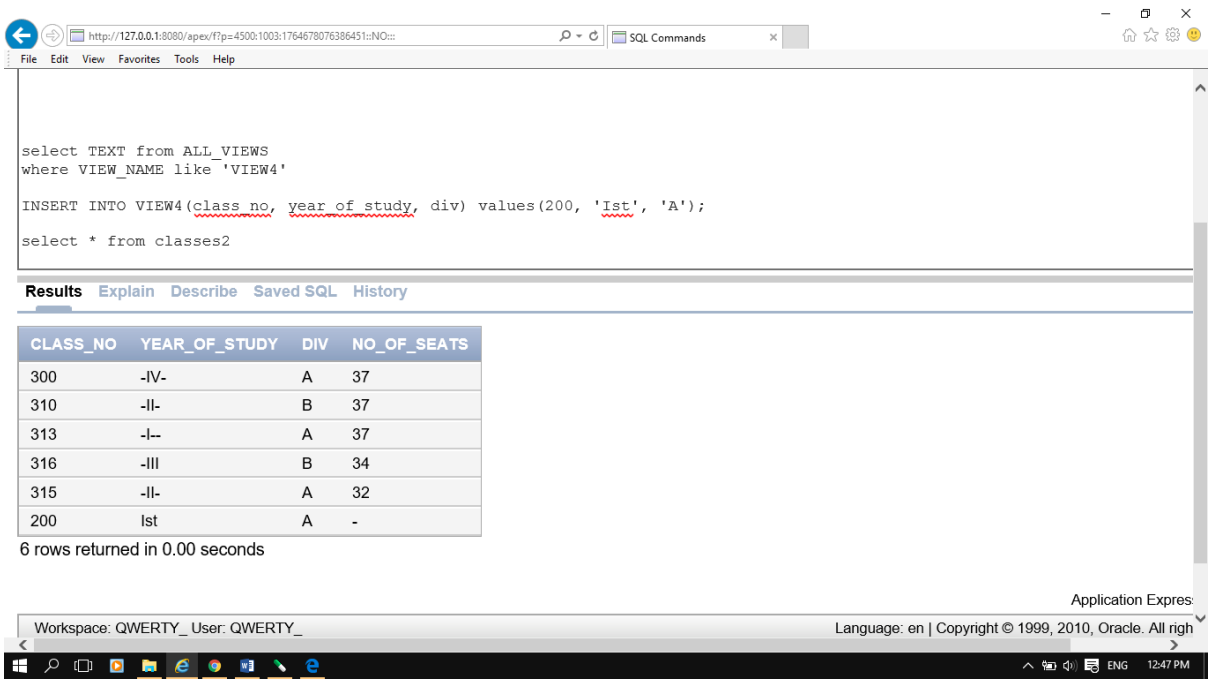
6 row(s) deleted.

select * from classes2

➔ No data found

Rollback
Statement processed

select * from classes2



drop table classes2
select * from view4
➔ Qwerty._view4 has errors

create table classes2 as select * from classes

select * from view4

CLASS_NO	YEAR_OF_STUDY	DIV
300	-IV-	A
310	-II-	B
313	-I--	A
316	-III	B
315	-II-	A

Design at least 10 SQL queries for suitable database application using SQL DML statements: Insert, Select, Update, Delete with operators, functions, and set operator

SELECT:

select * from student

ROLL_NO	FIRST_NAME	MIDDLE_NAME	LAST_NAME	MOBILE_NO	NATIVE_PLACE	DOB	CLASSROOM_NO
100007	Patricia	Patrick	D'gama	9823982311	Tiruvananthpuram, Kerala	01/10/1977	313
100008	Amruta	K	Raut	9823982311	Tiruvananthpuram, Kerala	01/10/1999	313
100009	Namrta	C	Pandey	9821111111	Tiruvanantpuram, "eral"	01/10/1999	313
100010	Kirti	R	Maheshwr	9823982311	Tiruvananthpuram, Kerala	01/10/1999	313
100001	Shreeya	Ajay	Chavan	9823198231	Pune	10/05/1999	316
100002	Sonal	A.	Chavan	9843198431	Pune	03/17/1998	310
100003	Shraddha	Bipen	Mehta	9443194431	Latur	09/05/1999	313
100004	Shruti	Satish	Salunke	9800098231	Aurangabad	11/05/1999	316
100005	Gauri	Santosh	Gad	9811198231	Goa	01/28/1999	316
100006	Shradha	Shailesh	Bhosale	9802190211	Mumbai	09/30/1999	300

create table places_school as select distinct(native_place) from student
select * from places_school

NATIVE_PLACE
Goa
Mumbai
Aurangabad
Tiruvananthpuram, Kerala
Tiruvanantpuram, "eral"
Pune
Latur

7 rows returned in 0.02 seconds

alter table places_school

add no_of_schools integer

```
Table altered.
```

0.23 seconds

---

update places_school

set no_of_schools = 10

where native_place in ('Goa', 'Mumbai', 'Pune')

➔
3 rows updated

update places_school

set no_of_schools = 20

where native_place like 'Aurang%' or native_place like 'Tiru%' or native_place like '%tur'
➔ 4 row(s) updated

select * from places_school

NATIVE_PLACE	NO_OF_SCHOOLS
Goa	10
Mumbai	10
Aurangabad	20

Tiruvananthpuram, Kerala	20
Tiruvanantpuram, "eral"	20
Pune	10
Latur	20

7 rows returned in 0.00 seconds

select distinct(student.native_place), placeS_school.no_of_schools from student, places_school where student.native_place = places_school.native_place;

NATIVE_PLACE	NO_OF_SCHOOLS
Pune	10
Mumbai	10
Latur	20
Aurangabad	20
Tiruvananthpuram, Kerala	20
Goa	10
Tiruvanantpuram, "eral"	20

7 rows returned in 0.06 seconds

---

create table salesman(salesman_id number(6), name varchar2(80), city varchar2(80), commission decimal(7, 2), constraint pk6 primary key(salesman_id));
->

```
Table created.
```

1.01 secondss

Write a SQL statement to display all the information of all salesmen.

insert into salesman(salesman_id, name, city, commission)

with NAMES as

(

SELECT 600001, 'John Hook', 'New York', 0.15 from dual union all

SELECT 600002, 'Nail Nite', 'London', 0.13 from dual union all

SELECT 600003, 'Pit Alex', 'Paris', 0.12 from dual union all

```
 SELECT 600004, 'Mc Lion', 'Paris',0.11 from dual union all

 SELECT 600005, 'Jony Hook', 'Rome', 0.10 from dual union all

 SELECT 600006, 'Tony Hong', 'San Jose', 0.16 from dual


)


select * from NAMES


select * from salesman
```

SALESMAN_ID	NAME	CITY	COMMISSION
600001	John Hook	New York	.15
600002	Nail Nite	London	.13
600003	Pit Alex	Paris	.12
600004	Mc Lion	Paris	.11
600005	Jony Hook	Rome	.1
600006	Tony Hong	San Jose	.16

```
declare

BEGIN

dbms_output.put_line('This is SQL Exercise, Practice and Solution.');

END;



select * from salesman

where commission =(select min(commission) from salesman)
```

SALESMAN_ID	NAME	CITY	COMMISSION
600005	Jony Hook	Rome	.1

select count(salesman_id) from salesman

COUNT(SALESMAN_ID)
6

select * from stud_inst

SID	INST_ID
100001	1
100002	2
100003	3
100004	4
100005	5
100006	6

select distinct(inst_id), count(sid) from stud_inst group by inst_id

INST_ID	COUNT(SID)
1	1
6	1
2	1
4	1
5	1
3	1

select distinct(inst_id),sid

from stud_inst

INST_ID	SID
1	100001
2	100002
4	100004
5	100005
6	100006
3	100003

6 rows returned in 0.01 seconds

select distinct(NATIVE_PLACE), count(roll_no) from student

group by native_place

NATIVE_PLACE	COUNT(ROLL_NO)
Goa	1
Mumbai	1
Aurangabad	1
Tiruvananthpuram, Kerala	3
Tiruvanantpuram, "eral"	1
Pune	2
Latur	1

7 rows returned in 0.00 seconds

select * from student

ROLL_NO	FIRST_NAME	MIDDLE_NAME	LAST_NAME	MOBILE_NO	NATIVE_PLACE	DOB	CLASSROOM_NO
100007	Patricia	Patrick	D'gama	9823982311	Tiruvananthpuram, Kerala	01/10/1977	313
100008	Amruta	K	Raut	9823982311	Tiruvananthpuram, Kerala	01/10/1999	313
100009	Namrta	C	Pandey	9821111111	Tiruvanantpuram, "eral"	01/10/1999	313
100010	Kirti	R	Maheshwr	9823982311	Tiruvananthpuram, Kerala	01/10/1999	313
100001	Shreeya	Ajay	Chavan	9823198231	Pune	10/05/1999	316
100002	Sonal	A.	Chavan	9843198431	Pune	03/17/1998	310
100003	Shraddha	Bipen	Mehta	9443194431	Latur	09/05/1999	313
100004	Shruti	Satish	Salunke	9800098231	Aurangabad	11/05/1999	316
100005	Gauri	Santosh	Gad	9811198231	Goa	01/28/1999	316
100006	Shradha	Shailesh	Bhosale	9802190211	Mumbai	09/30/1999	300

select * from student

where not native_place like '%nant%' and not native_place like '%ne'

ROLL_NO	FIRST_NAME	MIDDLE_NAME	LAST_NAME	MOBILE_NO	NATIVE_PLACE	DOB	CLASSROOM_NO
100003	Shraddha	Bipen	Mehta	9443194431	Latur	09/05/1999	313
100004	Shruti	Satish	Salunke	9800098231	Aurangabad	11/05/1999	316
100005	Gauri	Santosh	Gad	9811198231	Goa	01/28/1999	316
100006	Shradha	Shailesh	Bhosale	9802190211	Mumbai	09/30/1999	300

4 rows returned in 0.00 seconds

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

select roll_no, first_name from student

order by first_name DESC

ROLL_NO	FIRST_NAME
100002	Sonal
100004	Shruti
100001	Shreeya
100006	Shradha
100003	Shraddha
100007	Patricia
100009	Namrta
100010	Kirti
100005	Gauri
100008	Amruta

10 rows returned in 0.00 seconds

select roll_no, first_name from student

order by first_name ASC

ROLL_NO	FIRST_NAME
100008	Amruta

100005	Gauri
100010	Kirti
100009	Namrta
100007	Patricia
100003	Shraddha
100006	Shradha
100001	Shreeya
100004	Shruti
100002	Sonal

10 rows returned in 0.01 seconds

UPDATE STUDENT

SET FIRST_NAME = 'A', NATIVE_PLACE = 'b'

➔ 10 ROWS UPDATED

SELECT * FROM STUDENT

ROLL_NO	FIRST_NAME	MIDDLE_NAME	LAST_NAME	MOBILE_NO	NATIVE_PLACE	DOB	CLASSROOM_NO
100007	A	Patrick	D'gama	9823982311	b	01/10/1977	313
100008	A	K	Raut	9823982311	b	01/10/1999	313
100009	A	C	Pandey	9821111111	b	01/10/1999	313
100010	A	R	Maheshwr	9823982311	b	01/10/1999	313
100001	A	Ajay	Chavan	9823198231	b	10/05/1999	316
100002	A	A.	Chavan	9843198431	b	03/17/1998	310
100003	A	Bipen	Mehta	9443194431	b	09/05/1999	313
100004	A	Satish	Salunke	9800098231	b	11/05/1999	316
100005	A	Santosh	Gad	9811198231	b	01/28/1999	316
100006	A	Shailesh	Bhosale	9802190211	b	09/30/1999	300

rollback

SELECT * FROM STUDENT

➔ (Back to normal)

UPDATE STUDENT

SET FIRST_NAME = 'A' and NATIVE_PLACE = 'b'

```
ORA-00933: SQL command not properly ended
```

0.00 seconds

SYNONYM

create sequence seq2

minvalue 2

maxvalue 100

start with 2

increment by 1

CYCLE;

➔ Sequence created

select seq2.CURRVAL from DUAL;

select seq2.NEXTVAL from DUAL;

create table example(id integer)

➔ Table created

insert into example values(seq2.NEXTVAL)

alter sequence seq2

increment by -1

select * from example

order by id

ID
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36

37
38
39
40
41
42
43
44
45
46
47
48
48
49
49
50
50
51
51
52
52
53

56 rows returned in 0.00 seconds

SELECT ID, COUNT(ID) FROM EXAMPLE

GROUP BY ID

HAVING COUNT(ID) > 1

ORDER BY ID

ID	COUNT(ID)
48	2
49	2
50	2
51	2
52	2

select ROWID, ID FROM EXAMPLE

ROWID	ID
AAAFktAAEAAAAp9AAA	19

AAAFktAAEAAAAp9AAB	20
AAAFktAAEAAAAp9AAC	21
AAAFktAAEAAAAp9AAD	22
AAAFktAAEAAAAp9AAE	23
AAAFktAAEAAAAp9AAF	24
AAAFktAAEAAAAp9AAG	25
AAAFktAAEAAAAp9AAH	26
AAAFktAAEAAAAp9AAI	27
AAAFktAAEAAAAp9AAJ	28
AAAFktAAEAAAAp9AAK	29
AAAFktAAEAAAAp9AAL	30
AAAFktAAEAAAAp9AAM	31
AAAFktAAEAAAAp9AAN	32
AAAFktAAEAAAAp9AAO	33
AAAFktAAEAAAAp9AAP	34
AAAFktAAEAAAAp9AAQ	35
AAAFktAAEAAAAp9AAR	36
AAAFktAAEAAAAp9AAS	37
AAAFktAAEAAAAp9AAT	38
AAAFktAAEAAAAp9AAU	39
AAAFktAAEAAAAp9AAV	40
AAAFktAAEAAAAp9AAW	41
AAAFktAAEAAAAp9AAX	42
AAAFktAAEAAAAp9AAY	43
AAAFktAAEAAAAp9AAZ	44
AAAFktAAEAAAAp9AAa	45
AAAFktAAEAAAAp9AAb	46
AAAFktAAEAAAAp9AAc	47
AAAFktAAEAAAAp9AAd	48
AAAFktAAEAAAAp9AAe	49
AAAFktAAEAAAAp9AAf	50
AAAFktAAEAAAAp9AAg	51
AAAFktAAEAAAAp9AAh	52
AAAFktAAEAAAAp9AAi	53
AAAFktAAEAAAAp9AAj	52
AAAFktAAEAAAAp9AAk	51
AAAFktAAEAAAAp9AAl	50
AAAFktAAEAAAAp9AAm	49
AAAFktAAEAAAAp9AAn	48
AAAFktAAEAAAAp/AAA	3
AAAFktAAEAAAAp/AAB	4
AAAFktAAEAAAAp/AAC	5

AAAFktAAEAAAAp/AAD	6
AAAFktAAEAAAAp/AAE	7
AAAFktAAEAAAAp/AAF	8
AAAFktAAEAAAAp/AAG	9
AAAFktAAEAAAAp/AAH	10
AAAFktAAEAAAAp/AAI	11
AAAFktAAEAAAAp/AAJ	12
AAAFktAAEAAAAp/AAK	13
AAAFktAAEAAAAp/AAL	14
AAAFktAAEAAAAp/AAM	15
AAAFktAAEAAAAp/AAN	16
AAAFktAAEAAAAp/AAO	17
AAAFktAAEAAAAp/AAP	18

56 rows returned in 0.01 seconds

```
delete from example A
where ROWID> SELECT MIN(ROWID) FROM example
B
where B.id = A.id
```

error(brackets)

ORA-00936: missing expression

delete from example A

where ROWID> (SELECT MIN(ROWID) FROM example B

where B.id = A.id)

5 row(s) deleted

Design at least 10 SQL queries for suitable database application using SQL DML statements: Insert, Select, Update, Delete with operators, functions, and set operator

INSERT

SELECT

OPERATORS

https://docs.oracle.com/cd/B19188_01/doc/B15917/sqopr.htm

SELECT * FROM ALL_SEQUENCES

SEQUENCE_OWNER	SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	CYCLE_FLAG	ORDER_FLAG	CACHE_SIZE	LAST_NUMBER
SYS	SCHEDULER$_JOBSUFFIX_S	1	999999999999999999999999	1	N	N	20	1621
SYS	DM$EXPIMP_ID_SEQ	1	999999999999999999999999	1	N	N	20	1
SYS	HS_BULK_SEQ	1	999999999999999999999999	1	N	N	0	1
XDB	XDB$NAMESUFF_SEQ	1	99999	1	Y	N	20	401
MDSYS	SDO_IDX_TAB_SEQUENCE	1	999999999999999999999999	1	Y	N	20	1
MDSYS	SAMPLE_SEQ	1	999999999999999999999999	1	N	N	20	1
MDSYS	TMP_COORD_OPS	1000000	2000000	1	Y	N	0	1000000
APEX_040000	WWV_FLOW_SESSION_SEQ	1	999999999999999999999999	1	N	N	20	1
APEX_040000	WWV_SEQ	1	999999999999999999999999	1	N	N	20	63301
QWERTY_	DEMO_CUST_SEQ	1	999999999999999999999999	1	N	N	20	21
QWERTY_	DEMO_ORDER_ITEMS_SEQ	1	999999999999999999999999	1	N	N	20	61
QWERTY_	DEMO_ORD_SEQ	1	999999999999999999999999	1	N	N	20	11
QWERTY_	DEMO_PROD_SEQ	1	999999999999999999999999	1	N	N	20	21
QWERTY_	DEMO_USERS_SEQ	1	999999999999999999999999	1	N	N	20	21
QWERTY_	SEQ2	2	100	-1	Y	N	20	32

select * from ALL_TABLES

⇨ (All tables are shown)

UPDATE EXAMPLE A

SET ID = 100

WHERE ID IN (SELECT ID FROM EXAMPLE GROUP BY ID HAVING COUNT(ID) = 1)

51 row(s) updated

select * from example

ID
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100

100
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100
100

51 rows returned in 0.00 seconds

Rollback

Statement processed

select TEXT from ALL_VIEWS

WHERE VIEW_NAME like '%VIEW2%'

TEXT
(select student.roll_no, student.first_name, student.last_name, student.middle_name, instructor.inst_id, instructor.inst_first_name from student, instructor, stud_inst where student.roll_no = stud_inst.sid and instructor.inst_id = stud_inst.inst_id)

1 rows returned in 0.25 seconds

```
delete from example
where id  = 19
select count(*) from example
```

COUNT(*)
50

1 rows returned in 0.00 seconds

UNARY OPERATORS

+ operand

- Operand negates the operand

update example

set id = -id

⇨ 50 row(s) updated

select * from example

order by ID DESC

ID
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12

-13
-14
-15
-16
-17
-18
-20
-21
-22
-23
-24
-25
-26
-27
-28
-29
-30
-31
-32
-33
-34
-35
-36
-37
-38
-39
-40
-41
-42
-43
-44
-45
-46
-47
-48
-49
-50
-51
-52
-53

50 rows returned in 0.00 seconds

update example

set id = +id

50 row(s) updated

select * from example

order by ID DESC

----SAME AS ABOVE NO CHANGE ------

update example

set id = -id

------ RETAINED THE ORIGINAL POSITIVES BACK-------

alter table example add salary decimal(7,2)

table altered

create sequence seq3

MINVALUE 10000

MAXVALUE 100000

start with 10000

increment by 3000

CYCLE

Sequence created

alter table example

modify salary number(10)

Table altered

update example

set salary = seq3.NEXTVAL

50 row(s) updated

select * from example

ID	SALARY
20	10000
21	13000
22	16000
23	19000
24	22000
25	25000
26	28000
27	31000
28	34000
29	37000
30	40000
31	43000
32	46000
33	49000
34	52000
35	55000
36	58000
37	61000
38	64000
39	67000
40	70000
41	73000
42	76000
43	79000
44	82000
45	85000
46	88000
47	91000
48	94000
49	97000
50	100000
51	10000
52	13000
53	16000
3	19000
4	22000

5	25000
6	28000
7	31000
8	34000
9	37000
10	40000
11	43000
12	46000
13	49000
14	52000
15	55000
16	58000
17	61000
18	64000

50 rows returned in 0.01 seconds

INSERTING NULL VALUES

inseRt into example values(85, NULL)

update example

set salary = seq3.nextval

select * from example

ID	SALARY
20	34000
21	37000
22	40000
23	43000
24	46000
25	49000
26	52000
27	55000
28	58000
29	61000
30	64000
31	67000

32	70000
33	73000
34	76000
35	79000
36	82000
37	85000
38	88000
39	91000
40	94000
41	97000
42	100000
43	10000
44	13000
45	16000
46	19000
47	22000
48	25000
49	28000
50	31000
51	34000
52	37000
53	40000
3	43000
4	46000
5	49000
6	52000
7	55000
8	58000
9	61000
10	64000
11	67000
12	70000
13	73000
14	76000
15	79000
16	82000
17	85000
18	88000

50 rows returned in 0.00 seconds

update example

set salary = salary + 5

50 row(s) updated

update example

set salary = salary -10

50 row(s) updated

select id, salary/1000 from example

order by ID asc

ID	SALARY/1000
3	42.995
4	45.995
5	48.995
6	51.995
7	54.995
8	57.995
9	60.995
10	63.995
11	66.995
12	69.995
13	72.995
14	75.995
15	78.995
16	81.995
17	84.995
18	87.995
20	33.995
21	36.995
22	39.995
23	42.995
24	45.995
25	48.995
26	51.995
27	54.995
28	57.995
29	60.995
30	63.995
31	66.995
32	69.995

33	72.995
34	75.995
35	78.995
36	81.995
37	84.995
38	87.995
39	90.995
40	93.995
41	96.995
42	99.995
43	9.995
44	12.995
45	15.995
46	18.995
47	21.995
48	24.995
49	27.995
50	30.995
51	33.995
52	36.995
53	39.995

50 rows returned in 0.01 seconds

|| String concatenation operator

select 'THE SALARY IS '|| SALARY FROM EXAMPLE

'THESALARYIS'||SALARY
THE SALARY IS 33995
THE SALARY IS 36995
THE SALARY IS 39995
THE SALARY IS 42995
THE SALARY IS 45995
THE SALARY IS 48995
THE SALARY IS 51995
THE SALARY IS 54995
THE SALARY IS 57995
THE SALARY IS 60995
THE SALARY IS 63995
THE SALARY IS 66995

THE SALARY IS 69995
THE SALARY IS 72995
THE SALARY IS 75995
THE SALARY IS 78995
THE SALARY IS 81995
THE SALARY IS 84995
THE SALARY IS 87995
THE SALARY IS 90995
THE SALARY IS 93995
THE SALARY IS 96995
THE SALARY IS 99995
THE SALARY IS 9995
THE SALARY IS 12995
THE SALARY IS 15995
THE SALARY IS 18995
THE SALARY IS 21995
THE SALARY IS 24995
THE SALARY IS 27995
THE SALARY IS 30995
THE SALARY IS 33995
THE SALARY IS 36995
THE SALARY IS 39995
THE SALARY IS 42995
THE SALARY IS 45995
THE SALARY IS 48995
THE SALARY IS 51995
THE SALARY IS 54995
THE SALARY IS 57995
THE SALARY IS 60995
THE SALARY IS 63995
THE SALARY IS 66995
THE SALARY IS 69995
THE SALARY IS 72995
THE SALARY IS 75995
THE SALARY IS 78995
THE SALARY IS 81995
THE SALARY IS 84995
THE SALARY IS 87995

50 rows returned in 0.00 seconds

update example

set salary = salary + 5

select 'THE SALARY IS '|| ID || ' there ' || SALARY FROM EXAMPLE order by id asc

'THESALARYIS'||ID||'THERE'||SALARY
THE SALARY IS 3 there 43000
THE SALARY IS 4 there 46000
THE SALARY IS 5 there 49000
THE SALARY IS 6 there 52000
THE SALARY IS 7 there 55000
THE SALARY IS 8 there 58000
THE SALARY IS 9 there 61000
THE SALARY IS 10 there 64000
THE SALARY IS 11 there 67000
THE SALARY IS 12 there 70000
THE SALARY IS 13 there 73000
THE SALARY IS 14 there 76000
THE SALARY IS 15 there 79000
THE SALARY IS 16 there 82000
THE SALARY IS 17 there 85000
THE SALARY IS 18 there 88000
THE SALARY IS 20 there 34000
THE SALARY IS 21 there 37000
THE SALARY IS 22 there 40000
THE SALARY IS 23 there 43000
THE SALARY IS 24 there 46000
THE SALARY IS 25 there 49000
THE SALARY IS 26 there 52000
THE SALARY IS 27 there 55000
THE SALARY IS 28 there 58000
THE SALARY IS 29 there 61000
THE SALARY IS 30 there 64000
THE SALARY IS 31 there 67000
THE SALARY IS 32 there 70000
THE SALARY IS 33 there 73000
THE SALARY IS 34 there 76000
THE SALARY IS 35 there 79000

THE SALARY IS 36 there 82000
THE SALARY IS 37 there 85000
THE SALARY IS 38 there 88000
THE SALARY IS 39 there 91000
THE SALARY IS 40 there 94000
THE SALARY IS 41 there 97000
THE SALARY IS 42 there 100000
THE SALARY IS 43 there 10000
THE SALARY IS 44 there 13000
THE SALARY IS 45 there 16000
THE SALARY IS 46 there 19000
THE SALARY IS 47 there 22000
THE SALARY IS 48 there 25000
THE SALARY IS 49 there 28000
THE SALARY IS 50 there 31000
THE SALARY IS 51 there 34000
THE SALARY IS 52 there 37000
THE SALARY IS 53 there 40000

50 rows returned in 0.00 seconds

CANCAT FUNCTION

SELECT CONCAT(ID, ' HAS SALARY '), SALARY FROM EXAMPLE

CONCAT(ID,'HASSALARY')	SALARY
20 HAS SALARY	34000
21 HAS SALARY	37000
22 HAS SALARY	40000
23 HAS SALARY	43000
24 HAS SALARY	46000
25 HAS SALARY	49000
26 HAS SALARY	52000
27 HAS SALARY	55000
28 HAS SALARY	58000
29 HAS SALARY	61000
30 HAS SALARY	64000
31 HAS SALARY	67000
32 HAS SALARY	70000
33 HAS SALARY	73000
34 HAS SALARY	76000
35 HAS SALARY	79000
36 HAS SALARY	82000

37 HAS SALARY	85000
38 HAS SALARY	88000
39 HAS SALARY	91000
40 HAS SALARY	94000
41 HAS SALARY	97000
42 HAS SALARY	100000
43 HAS SALARY	10000
44 HAS SALARY	13000
45 HAS SALARY	16000
46 HAS SALARY	19000
47 HAS SALARY	22000
48 HAS SALARY	25000
49 HAS SALARY	28000
50 HAS SALARY	31000
51 HAS SALARY	34000
52 HAS SALARY	37000
53 HAS SALARY	40000
3 HAS SALARY	43000
4 HAS SALARY	46000
5 HAS SALARY	49000
6 HAS SALARY	52000
7 HAS SALARY	55000
8 HAS SALARY	58000
9 HAS SALARY	61000
10 HAS SALARY	64000
11 HAS SALARY	67000
12 HAS SALARY	70000
13 HAS SALARY	73000
14 HAS SALARY	76000
15 HAS SALARY	79000
16 HAS SALARY	82000
17 HAS SALARY	85000
18 HAS SALARY	88000

50 rows returned in 0.01 seconds

select * from ALL_SEQUENCES where

SEQUENCE_OWNER LIKE '%QWERTY_%'

⇨

SEQUENCE_OWNER	SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	CYCLE_FLAG	ORDER_FLAG	CACHE_SIZE	LAST_NUMBER
QWERTY_	DEMO_CUST_SEQ	1	9999999999999999999999999	1	N	N	20	21
QWERTY_	DEMO_ORDER_ITEMS_SEQ	1	9999999999999999999999999	1	N	N	20	61
QWERTY_	DEMO_ORD_SEQ	1	9999999999999999999999999	1	N	N	20	11
QWERTY_	DEMO_PROD_SEQ	1	9999999999999999999999999	1	N	N	20	21
QWERTY_	DEMO_USERS_SEQ	1	9999999999999999999999999	1	N	N	20	21
QWERTY_	SEQ2	2	100	-1	Y	N	20	32
QWERTY_	SEQ3	10000	100000	3000	Y	N	20	70000

7 rows returned in 0.00 seconds

update example

set salary = seq3.NEXTVAL

⇨ 50 row(s) updated

⇨ ID	SALARY
20	13000
21	16000
22	19000
23	22000
24	25000
25	28000
26	31000
27	34000
28	37000
29	40000
30	43000
31	46000
32	49000
33	52000
34	55000

35	58000
36	61000
37	64000
38	67000
39	70000
40	73000
41	76000
42	79000
43	82000
44	85000
45	88000
46	91000
47	94000
48	97000
49	100000
50	10000
51	13000
52	16000
53	19000
3	22000
4	25000
5	28000
6	31000
7	34000
8	37000
9	40000
10	43000
11	46000
12	49000
13	52000
14	55000
15	58000
16	61000
17	64000
18	67000

50 rows returned in 0.00 seconds

https://beginner-sql-tutorial.com/oracle-functions.htm

What is a DUAL Table in Oracle?

This is a single row and single column dummy table provided by oracle. This is used to perform mathematical calculations without using a table.

NUMERIC FUNCTIONS

select ABS(-10) from DUAL

ABS(-10)
10

1 rows returned in 0.00 seconds

select CEIL(10.50) from DUAL

CEIL(10.50)
11

1 rows returned in 0.00 seconds

select FLOOR(10.50) from DUAL

FLOOR(10.50)
10

1 rows returned in 0.00 seconds

select ROUND(10.40) from DUAL

ROUND(10.40)
10

1 rows returned in 0.00 seconds

ROUND(125.456,-1)
130

1 rows returned in 0.00 seconds

select TRUNC (140.234, 2) from DUAL

TRUNC(140.234,2)
140.23

1 rows returned in 0.00 seconds

CHARACTER OR TEXT FUNCTIONS

Select * from student

ROLL_NO	FIRST_NAME	MIDDLE_NAME	LAST_NAME	MOBILE_NO	NATIVE_PLACE	DOB	CLASSROOM_NO
100007	Patricia	Patrick	D'gama	9823982311	Tiruvananthpuram, Kerala	01/10/1977	313
100008	Amruta	K	Raut	9823982311	Tiruvananthpuram, Kerala	01/10/1999	313
100009	Namrta	C	Pandey	9821111111	Tiruvanantpuram, "eral"	01/10/1999	313
100010	Kirti	R	Maheshwr	9823982311	Tiruvananthpuram, Kerala	01/10/1999	313
100001	Shreeya	Ajay	Chavan	9823198231	Pune	10/05/1999	316
100002	Sonal	A.	Chavan	9843198431	Pune	03/17/1998	310
100003	Shraddha	Bipen	Mehta	9443194431	Latur	09/05/1999	313
100004	Shruti	Satish	Salunke	9800098231	Aurangabad	11/05/1999	316
100005	Gauri	Santosh	Gad	9811198231	Goa	01/28/1999	316
100006	Shradha	Shailesh	Bhosale	9802190211	Mumbai	09/30/1999	300

10 rows returned in 0.00 seconds

select UPPER(first_name) from student

UPPER(FIRST_NAME)
PATRICIA
AMRUTA
NAMRTA
KIRTI
SHREEYA
SONAL
SHRADDHA
SHRUTI
GAURI
SHRADHA

10 rows returned in 0.03 seconds

select LOWER(first_name) from student

LOWER(FIRST_NAME)
patricia
amruta
namrta
kirti
shreeya
sonal
shraddha
shruti
gauri
shradha

10 rows returned in 0.00 seconds

select INITCAP(first_name) || ' ' || INITCAP(LAST_NAME) from student

INITCAP(FIRST_NAME)||''||INITCAP(LAST_NAME)
Patricia D'Gama
Amruta Raut
Namrta Pandey
Kirti Maheshwr
Shreeya Chavan
Sonal Chavan
Shraddha Mehta
Shruti Salunke
Gauri Gad
Shradha Bhosale

select LTRIM(INITCAP(first_name) || ' ' || INITCAP(LAST_NAME), 'S') from student

where First_name like 'S%'

LTRIM(INITCAP(FIRST_NAME)||''||INITCAP(LAST_NAME),'S')
hreeya Chavan
onal Chavan
hraddha Mehta
hruti Salunke
hradha Bhosale

5 rows returned in 0.00 seconds

select LTRIM(INITCAP(first_name) || ' ' || INITCAP(LAST_NAME), 'S'), LENGTH(LTRIM(INITCAP(first_name) || ' ' || INITCAP(LAST_NAME), 'S')) from student

where First_name like 'S%'

LTRIM(INITCAP(FIRST_NAME)||''||INITCAP(LAST_NAME),'S')	LENGTH(LTRIM(INITCAP(FIRST_NAME)||''||INITCAP(LAST_NAME),'S'))
hreeya Chavan	13
onal Chavan	11
hraddha Mehta	13
hruti Salunke	13
hradha Bhosale	14

5 rows returned in 0.01 seconds

select RTRIM(INITCAP(first_name) || ' ' || INITCAP(LAST_NAME), 'n'), LENGTH(RTRIM(INITCAP(first_name) || ' ' || INITCAP(LAST_NAME), 'n')) from student

where last_name like '%n'

RTRIM(INITCAP(FIRST_NAME)||''||INITCAP(LAST_NAME),'N')	LENGTH(RTRIM(INITCAP(FIRST_NAME)||''||INITCAP(LAST_NAME),'N'))
Shreeya Chava	13
Sonal Chava	11

2 rows returned in 0.00 seconds

select ID, LPAD(SALARY, 10, '_')FROM EXAMPLE

ID	LPAD(SALARY,10,'_')
20	_____13000
21	_____16000
22	_____19000
23	_____22000
24	_____25000
25	_____28000
26	_____31000
27	_____34000
28	_____37000
29	_____40000
30	_____43000

31	_____46000
32	_____49000
33	_____52000
34	_____55000
35	_____58000
36	_____61000
37	_____64000
38	_____67000
39	_____70000
40	_____73000
41	_____76000
42	_____79000
43	_____82000
44	_____85000
45	_____88000
46	_____91000
47	_____94000
48	_____97000
49	____100000
50	_____10000
51	_____13000
52	_____16000
53	_____19000
3	_____22000
4	_____25000
5	_____28000
6	_____31000
7	_____34000
8	_____37000
9	_____40000
10	_____43000
11	_____46000
12	_____49000
13	_____52000
14	_____55000
15	_____58000
16	_____61000
17	_____64000
18	_____67000

50 rows returned in 0.00 seconds

select ID, RPAD(SALARY, 10, '_')FROM EXAMPLE

order by ID

DATE FUNCTIONS

select ROLL_NO, FIRST_NAME || ' '|| MIDDLE_NAME || ' ' || LAST_NAME DOB, MONTHS_BETWEEN (SYSDATE, DOB) FROM STUDENT

ROLL_NO	DOB	MONTHS_BETWEEN(SYSDATE,DOB)
100007	Patricia Patrick D'gama	513.31193324372759856630824372759856630 8
100008	Amruta K Raut	249.31193324372759856630824372759856630 8
100009	Namrta C Pandey	249.31193324372759856630824372759856630 8
100010	Kirti R Maheshwr	249.31193324372759856630824372759856630 8
100001	Shreeya Ajay Chavan	240.47322356630824372759856630824372759 9
100002	Sonal A. Chavan	259.08612679211469534050179211469534050 2
100003	Shraddha Bipen Mehta	241.47322356630824372759856630824372759 9
100004	Shruti Satish Salunke	239.47322356630824372759856630824372759 9
100005	Gauri Santosh Gad	248.73128808243727598566308243727598566 3
100006	Shradha Shailesh Bhosale	240.66677195340501792114695340501792114 7

10 rows returned in 0.00 seconds

Select NEXT_DAY(SYSDATE, 'TUESDAY') from dual

NEXT_DAY(SYSDATE,'TUESDAY')
10/22/2019

1 rows returned in 0.00 seconds

SQL SUBQUERY

https://beginner-sql-tutorial.com/sql-subquery.htm

select * from student

where first_name like 'Shre%'

UNION

select * from student

where first_name like 'Sonal%'

ROLL_NO	FIRST_NAME	MIDDLE_NAME	LAST_NAME	MOBILE_NO	NATIVE_PLACE	DOB	CLASSROOM_NO
100001	Shreeya	Ajay	Chavan	9823198231	Pune	10/05/1999	316
100002	Sonal	A.	Chavan	9843198431	Pune	03/17/1998	310

2 rows returned in 0.00 seconds

create table student2 as select * from student

Table created

select * from student

UNION select * from student2

- ⇨ (10 row(s) are returned)

update student2

set first_name = 'Miss.' || first_name

where first_name like 'S%'

- ⇨ 5 row(s) updated

select * from student

INTERSECT select * from student2

ROLL_NO	FIRST_NAME	MIDDLE_NAME	LAST_NAME	MOBILE_NO	NATIVE_PLACE	DOB	CLASSROOM_NO
100005	Gauri	Santosh	Gad	9811198231	Goa	01/28/1999	316
100007	Patricia	Patrick	D'gama	9823982311	Tiruvananthpuram, Kerala	01/10/1977	313
100008	Amruta	K	Raut	9823982311	Tiruvananthpuram, Kerala	01/10/1999	313
100009	Namrta	C	Pandey	9821111111	Tiruvanantpuram, "eral"	01/10/1999	313
100010	Kirti	R	Maheshwr	9823982311	Tiruvananthpuram, Kerala	01/10/1999	313

5 rows returned in 0.01 seconds

select * from student

MINUS select * from student2

ROLL_NO	FIRST_NAME	MIDDLE_NAME	LAST_NAME	MOBILE_NO	NATIVE_PLACE	DOB	CLASSROOM_NO
100001	Shreeya	Ajay	Chavan	9823198231	Pune	10/05/1999	316
100002	Sonal	A.	Chavan	9843198431	Pune	03/17/1998	310
100003	Shraddha	Bipen	Mehta	9443194431	Latur	09/05/1999	313
100004	Shruti	Satish	Salunke	9800098231	Aurangabad	11/05/1999	316
100006	Shradha	Shailesh	Bhosale	9802190211	Mumbai	09/30/1999	300

5 rows returned in 0.00 seconds

EXPERIMENT No. 3

Design at least 10 SQL queries for suitable database application using SQL DML statements: all types of Join, Sub-Query and View.

http://www.sql-join.com/sql-join-types

Use of SQL Join

There is Physical Join(Implementing physical relation between two tables using integrity constraints)

Logical Join

1. Combined Data with set operators

2. Combinational Data

Cross join, equi join, inner join, self join, outer join divided into 3 – left, right, full

Cross join includes all possible combinations valid and invalid.

create table instructor_classes(inst_id number(6), degree varchar2(80), constraint fk1 foreign key(inst_id) references instructor(inst_id));

⇨ Table created

create sequence seq5

start with 1

MINVALUE 1

INCREMENT BY 1

MAXVALUE 1000

NOCYCLE

- ⇨ Sequence created

update instructor_Classes

set inst_id = seq5.NEXTVAL

alter table instructor_classes

modify inst_id number(6) NOT NULL

select * from instructor A inner join instructor_classes B

on A.inst_id = B.inst_id

INST_ID	INST_FIRST_NAME	INST_MIDDLE_NAME	INST_LAST_NAME	YEAR_OF_JOINING	INST_SALARY	INST_ID	DEGREE
1	Ajay	V.	Chavan	2014	10000	1	M.E.
2	Vijay	V.	Bhosale	2014	10000	2	M.E.
3	Sujay	V.	Salunke	2014	10000	3	M.E.
4	Dhananjay	V.	Paiyawal	2014	10000	4	M.E.
5	MrityunAjay	V.	Panchbhai	2014	10000	5	B.E.
6	Jay	V.	Ramdasi	2014	10000	6	B.E.
7	Sanjay	V.	Iyer	2014	10000	7	B.E.
8	Kiran	V.	Mehta	2014	10000	8	B.E.
9	Naveen	S.	Reddy	2014	10000	9	PHD

10	Sanchit	R.	Jain	2014	10000	10	PHD
11	Corey	M.	Schafer	2014	10000	11	PHD

11 rows returned in 0.06 seconds

select * from instructor A left join instructor_classes B

on A.inst_id = B.inst_id

- ⇨ Same output as above

select * from instructor A right join instructor_classes B

on A.inst_id = B.inst_id

- ⇨ Same as above

select * from instructor A left join instructor_classes B

on A.inst_id = B.inst_id

- ⇨ SAME AS ABOVE

select * from instructor A full outer join instructor_classes B

on A.inst_id = B.inst_id

- ⇨ Same as above

select * from instructor A join instructor B

on A.inst_id = B.inst_id

=>

INST_ID	INST_FIRST_NAME	INST_MIDDLE_NAME	INST_LAST_NAME	YEAR_OF_JOINING	INST_SALARY	INST_ID	INST_FIRST_NAME	INST_MIDDLE_NAME	INST_LAST_NAME	YEAR_OF_JOINING	INST_SALARY
1	Ajay	V.	Chavan	2014	10000	1	Ajay	V.	Chavan	2014	10000
2	Vijay	V.	Bhosale	2014	10000	2	Vijay	V.	Bhosale	2014	10000
3	Sujay	V.	Salunke	2014	10000	3	Sujay	V.	Salunke	2014	10000
4	Dhananjay	V.	Paiyawal	2014	10000	4	Dhananjay	V.	Paiyawal	2014	10000

5	MrityunAjay	V.	Panchbhai	2014	10000	5	MrityunAjay	V.	Panchbhai	2014	10000
6	Jay	V.	Ramdasi	2014	10000	6	Jay	V.	Ramdasi	2014	10000
7	Sanjay	V.	Iyer	2014	10000	7	Sanjay	V.	Iyer	2014	10000
8	Kiran	V.	Mehta	2014	10000	8	Kiran	V.	Mehta	2014	10000
9	Naveen	S.	Reddy	2014	10000	9	Naveen	S.	Reddy	2014	10000
10	Sanchit	R.	Jain	2014	10000	10	Sanchit	R.	Jain	2014	10000
11	Corey	M.	Schafer	2014	10000	11	Corey	M.	Schafer	2014	10000

select instructor.inst_id, instructor_classes.inst_id, inst_first_name || ' ' || inst_last_name, degree

from instructor, instructor_classes

INST_ID	INST_ID	INST_FIRST_NAME||''||INST_LAST_NAME	DEGREE
1	1	Ajay Chavan	M.E.
1	2	Ajay Chavan	M.E.
1	3	Ajay Chavan	M.E.
1	4	Ajay Chavan	M.E.
1	5	Ajay Chavan	B.E.
1	6	Ajay Chavan	B.E.
1	7	Ajay Chavan	B.E.
1	8	Ajay Chavan	B.E.
1	9	Ajay Chavan	PHD
1	10	Ajay Chavan	PHD
1	11	Ajay Chavan	PHD
2	1	Vijay Bhosale	M.E.
2	2	Vijay Bhosale	M.E.
2	3	Vijay Bhosale	M.E.
2	4	Vijay Bhosale	M.E.
2	5	Vijay Bhosale	B.E.
2	6	Vijay Bhosale	B.E.
2	7	Vijay Bhosale	B.E.
2	8	Vijay Bhosale	B.E.
2	9	Vijay Bhosale	PHD
2	10	Vijay Bhosale	PHD
2	11	Vijay Bhosale	PHD
3	1	Sujay Salunke	M.E.
3	2	Sujay Salunke	M.E.
3	3	Sujay Salunke	M.E.
3	4	Sujay Salunke	M.E.
3	5	Sujay Salunke	B.E.
3	6	Sujay Salunke	B.E.
3	7	Sujay Salunke	B.E.
3	8	Sujay Salunke	B.E.
3	9	Sujay Salunke	PHD
3	10	Sujay Salunke	PHD
3	11	Sujay Salunke	PHD

4	1	Dhananjay Paiyawal	M.E.
4	2	Dhananjay Paiyawal	M.E.
4	3	Dhananjay Paiyawal	M.E.
4	4	Dhananjay Paiyawal	M.E.
4	5	Dhananjay Paiyawal	B.E.
4	6	Dhananjay Paiyawal	B.E.
4	7	Dhananjay Paiyawal	B.E.
4	8	Dhananjay Paiyawal	B.E.
4	9	Dhananjay Paiyawal	PHD
4	10	Dhananjay Paiyawal	PHD
4	11	Dhananjay Paiyawal	PHD
5	1	MrityunAjay Panchbhai	M.E.
5	2	MrityunAjay Panchbhai	M.E.
5	3	MrityunAjay Panchbhai	M.E.
5	4	MrityunAjay Panchbhai	M.E.
5	5	MrityunAjay Panchbhai	B.E.
5	6	MrityunAjay Panchbhai	B.E.
5	7	MrityunAjay Panchbhai	B.E.
5	8	MrityunAjay Panchbhai	B.E.
5	9	MrityunAjay Panchbhai	PHD
5	10	MrityunAjay Panchbhai	PHD
5	11	MrityunAjay Panchbhai	PHD
6	1	Jay Ramdasi	M.E.
6	2	Jay Ramdasi	M.E.
6	3	Jay Ramdasi	M.E.
6	4	Jay Ramdasi	M.E.
6	5	Jay Ramdasi	B.E.
6	6	Jay Ramdasi	B.E.
6	7	Jay Ramdasi	B.E.
6	8	Jay Ramdasi	B.E.
6	9	Jay Ramdasi	PHD
6	10	Jay Ramdasi	PHD
6	11	Jay Ramdasi	PHD
7	1	Sanjay Iyer	M.E.
7	2	Sanjay Iyer	M.E.
7	3	Sanjay Iyer	M.E.
7	4	Sanjay Iyer	M.E.
7	5	Sanjay Iyer	B.E.
7	6	Sanjay Iyer	B.E.
7	7	Sanjay Iyer	B.E.
7	8	Sanjay Iyer	B.E.
7	9	Sanjay Iyer	PHD
7	10	Sanjay Iyer	PHD
7	11	Sanjay Iyer	PHD
8	1	Kiran Mehta	M.E.
8	2	Kiran Mehta	M.E.
8	3	Kiran Mehta	M.E.
8	4	Kiran Mehta	M.E.
8	5	Kiran Mehta	B.E.
8	6	Kiran Mehta	B.E.
8	7	Kiran Mehta	B.E.
8	8	Kiran Mehta	B.E.
8	9	Kiran Mehta	PHD

8	10	Kiran Mehta	PHD
8	11	Kiran Mehta	PHD
9	1	Naveen Reddy	M.E.
9	2	Naveen Reddy	M.E.
9	3	Naveen Reddy	M.E.
9	4	Naveen Reddy	M.E.
9	5	Naveen Reddy	B.E.
9	6	Naveen Reddy	B.E.
9	7	Naveen Reddy	B.E.
9	8	Naveen Reddy	B.E.
9	9	Naveen Reddy	PHD
9	10	Naveen Reddy	PHD
9	11	Naveen Reddy	PHD
10	1	Sanchit Jain	M.E.
10	2	Sanchit Jain	M.E.
10	3	Sanchit Jain	M.E.
10	4	Sanchit Jain	M.E.
10	5	Sanchit Jain	B.E.
10	6	Sanchit Jain	B.E.
10	7	Sanchit Jain	B.E.
10	8	Sanchit Jain	B.E.
10	9	Sanchit Jain	PHD
10	10	Sanchit Jain	PHD
10	11	Sanchit Jain	PHD
11	1	Corey Schafer	M.E.
11	2	Corey Schafer	M.E.
11	3	Corey Schafer	M.E.
11	4	Corey Schafer	M.E.
11	5	Corey Schafer	B.E.
11	6	Corey Schafer	B.E.
11	7	Corey Schafer	B.E.
11	8	Corey Schafer	B.E.
11	9	Corey Schafer	PHD
11	10	Corey Schafer	PHD
11	11	Corey Schafer	PHD

121 rows returned in 0.01 seconds

select instructor.inst_id, instructor_classes.inst_id, inst_first_name || ' ' || inst_last_name, degree from instructor, instructor_classes where instructor.inst_id = instructor_classes.inst_id

INST_ID	INST_ID	INST_FIRST_NAME||''||INST_LAST_NAME	DEGREE
1	1	Ajay Chavan	M.E.
2	2	Vijay Bhosale	M.E.
3	3	Sujay Salunke	M.E.
4	4	Dhananjay Paiyawal	M.E.
5	5	MrityunAjay Panchbhai	B.E.
6	6	Jay Ramdasi	B.E.
7	7	Sanjay Iyer	B.E.
8	8	Kiran Mehta	B.E.
9	9	Naveen Reddy	PHD

10	10	Sanchit Jain	PHD
11	11	Corey Schafer	PHD

11 rows returned in 0.00 seconds

RESETTING THE SEQUENCE

So, by finding out the current value of the sequence and altering the increment by to be negative that number and selecting the sequence once -- we can reset it. Just beware that if others are using the sequence during this time - they (or you) may get

ORA-08004: sequence SEQ.NEXTVAL goes below MINVALUE and cannot be instantiated

until you set the sequence increment back to +1.

This would be preferred to dropping and recreating the sequence which would invalidate any dependent objects (such as triggers/stored procedures and so on)

https://asktom.oracle.com/pls/asktom/f?p=100:11:0::::P11_QUESTION_ID:1119633817597

AMBIGUOUS COLUMS

select inst_id, inst_first_name || ' ' || inst_last_name, degree

from instructor, instructor_classes

ORA-00918: column ambiguously defined

Solution: Use Tablename.common_column name or use alias

select instructor.inst_id, instructor_classes.inst_id, inst_first_name || ' ' || inst_last_name, degree

from instructor, instructor_classes

OR

select A.inst_id, B.inst_id, A.inst_first_name || ' ' || A.inst_last_name, B.degree

from instructor A,

instructor_classes B

But some combinations are invalid

Unnamed PL/SQL code block: Use of Control structure and Exception handling is mandatory. Write a PL/SQL block of code for Insert, Update and Delete

DECLARE

x number;

BEGIN

dbms_output.put_line('HII DEAR');

END;

```
Output: HII DEAR
```

- ⇨ Statement processed.

Accepting User Input

DECLARE

nn number;

BEGIN

nn := :nn;

dbms_output.put_line(nn);

END;

DECLARE

roll_no1 student.ROLL_NO%TYPE;

FIRST_NAME2 student.FIRST_NAME%TYPE;

```
LAST_NAME1 student.LAST_NAME%TYPE;

MOBILE_NO1 student.MOBILE_NO%TYPE;

NATIVE_PLACE1 student.NATIVE_PLACE%TYPE;

DOB1 student.DOB%TYPE;

CLASSROOM_NO1 student.CLASSROOM_NO%TYPE;

N number;

BEGIN

DBMS_OUTPUT.PUT_LINE('Enter the number of rows you wanna insert: ');

N := :Number_of_rows;

WHILE N>=1 LOOP

roll_no1 := :roll_no;

FIRST_NAME2:= :FIRST_NAMe;

LAST_NAME1:= :LAST_NAME;
```

```
MOBILE_NO1:= :MOBILE_NO;


NATIVE_PLACE1:= :NATIVE_PLACE;


DOB1 := :DOB;


CLASSROOM_NO1 := :CLASSROOM_NO;

dbms_output.put_line(roll_no1 || FIRST_NAME2 || LAST_NAME1 || MOBILE_NO1 ||
NATIVE_PLACE1 || DOB1 || CLASSROOM_NO1);


insert into student(ROLL_NO, FIRST_NAME, MIDDLE_NAME, LAST_NAME, MOBILE_NO,
NATIVE_PLACE, DOB, CLASSROOM_NO) values(roll_no1, FIRST_NAME2, 'SONKIRAN',
LAST_NAME1, MOBILE_NO1, NATIVE_PLACE1, DOB1, CLASSROOM_NO1);


N:= N-1;

END LOOP;


END;




select * from student


alter table student

add percentage number(3);




create sequence seq6

start with 70

MINVALUE 70
```

MAXVALUE 100

INCREMENT BY 1

CYCLE;

update student

set percentage = seq6.NEXTVAL;

select * from student;

ROLL_NO	FIRST_NAME	MIDDLE_NAME	LAST_NAME	MOBILE_NO	NATIVE_PLACE	DOB	CLASSROOM_NO	PERCENTAGE
100007	Patricia	Patrick	D'gama	9823982311	Tiruvananthpuram, Kerala	01/10/1977	313	70
100008	Amruta	K	Raut	9823982311	Tiruvananthpuram, Kerala	01/10/1999	313	71
100009	Namrta	C	Pandey	9821111111	Tiruvanantpuram, "eral"	01/10/1999	313	72
100010	Kirti	R	Maheshwr	9823982311	Tiruvananthpuram, Kerala	01/10/1999	313	73
100001	Shreeya	Ajay	Chavan	9823198231	Pune	10/05/1999	316	74
100002	Sonal	A.	Chavan	9843198431	Pune	03/17/1998	310	75
100003	Shraddha	Bipen	Mehta	9443194431	Latur	09/05/1999	313	76
100004	Shruti	Satish	Salunke	9800098231	Aurangabad	11/05/1999	316	77
100005	Gauri	Santosh	Gad	9811198231	Goa	01/28/1999	316	78
100006	Shradha	Shailesh	Bhosale	9802190211	Mumbai	09/30/1999	300	79

10 rows returned in 0.00 seconds

seleCt ROLL_NO, FIRST_NAME, MIDDLE_NAME, LAST_NAME, PERCENTAGE, (CASE

WHEN PERCENTAGE <72 THEN 'FAIL' WHEN PERCENTAGE <75 THEN 'B' WHEN PERCENTAGE < 80 THEN 'A' ELSE 'POOR' END) "GRADE" FROM STUDENT

ROLL_NO	FIRST_NAME	MIDDLE_NAME	LAST_NAME	PERCENTAGE	GRADE
100007	Patricia	Patrick	D'gama	70	FAIL
100008	Amruta	K	Raut	71	FAIL
100009	Namrta	C	Pandey	72	B
100010	Kirti	R	Maheshwr	73	B
100001	Shreeya	Ajay	Chavan	74	B
100002	Sonal	A.	Chavan	75	A
100003	Shraddha	Bipen	Mehta	76	A
100004	Shruti	Satish	Salunke	77	A
100005	Gauri	Santosh	Gad	78	A
100006	Shradha	Shailesh	Bhosale	79	A

10 rows returned in 0.00 seconds

https://docs.oracle.com/cd/A58617_01/server.804/a58312/newch232.htm

ERROR MESSAGES

create sequence sequence7

START WITH 101001

MINVALUE 101001

INCREMENT BY 1

NOCYCLE

```
DECLARE

roll_no1 student.ROLL_NO%TYPE;

FIRST_NAME2 student.FIRST_NAME%TYPE;

MIDDLE_NAME1 student.MIDDLE_NAME%TYPE;
LAST_NAME1 student.LAST_NAME%TYPE;

MOBILE_NO1 student.MOBILE_NO%TYPE;

NATIVE_PLACE1 student.NATIVE_PLACE%TYPE;

DOB1 student.DOB%TYPE;

CLASSROOM_NO1 student.CLASSROOM_NO%TYPE;

N number;


BEGIN

DBMS_OUTPUT.PUT_LINE('Enter the number of rows you wanna insert: ');
```

```
N := :Number_of_rows;

WHILE N>=1 LOOP

roll_no1 := :roll_no;

FIRST_NAME2:= :FIRST_NAMe;

MIDDLE_NAME1 := :MIDDLE;

LAST_NAME1:= :LAST_NAME;

MOBILE_NO1:= :MOBILE_NO;

NATIVE_PLACE1:= :NATIVE_PLACE;

DOB1 := :DOB;

CLASSROOM_NO1 := :CLASSROOM_NO;
dbms_output.put_line(roll_no1 || FIRST_NAME2 || LAST_NAME1 || MOBILE_NO1 ||
NATIVE_PLACE1 || DOB1 || CLASSROOM_NO1);

insert into student(ROLL_NO, FIRST_NAME, MIDDLE_NAME, LAST_NAME, MOBILE_NO,
NATIVE_PLACE, DOB, CLASSROOM_NO, PERCENTAGE) values(roll_no1, FIRST_NAME2,
MIDDLE_NAME1, LAST_NAME1, MOBILE_NO1, NATIVE_PLACE1, DOB1, CLASSROOM_NO1,
SEQ6.NEXTVAL);

N:= N-1;
END LOOP;

END;
```

Unnamed PL/SQL code block: Use of Control structure and Exception handling is mandatory. Write a PL/SQL block of code for the following requirements:- Schema: 1. Borrower(Rollin, Name, DateofIssue, NameofBook, Status) 2. Fine(Roll_no,Date,Amt) Accept roll_no & name of book from user. Check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs 5per day. If no. of days>30, per day fine will be Rs 50 per day & for days less than 30, Rs. 5 per day. After submitting the book, status will change from I to R. If condition of fine is true, then details will be stored into fine table. Frame the problem statement for writing PL/SQL block inline with above statement

create table Borrower(ROLL_NO number(10), DateOfIssue date, NameOfBook varchar2(80), constraint fk1212 foreign key(roll_no) references student(roll_no));

⇨ Table created

alter table borrower
add status char(1);

⇨ Table altered

create table Fine(ROLL_NO number(10), Date1 date default(SYSDATE), Amt number(10));

⇨ Table created

insert into borrower values(100005, '04-21-2019', 'Theory Of Computation', 'I');
…..
….
….
Inserting values

Select * from borrower;

ROLL_NO	DATEOFISSUE	NAMEOFBOOK	STATUS
100005	04/21/2019	Theory Of Computation	I
100001	10/05/2019	Shatranj ke Khiladi	I
100002	10/07/2019	DATABASE SYSTEM CONCEPTS	I
100003	10/01/2019	DATA COMMUNICATIONS AND NW	I
100004	09/30/2019	DOLLAR BAHU	I

```
DECLARE
t_rollno BORROWER.ROLL_NO%TYPE;
t_nameofbook BORROWER.NAMEOFBOOK%TYPE;
ISSUEDATE date;
TOTAL_FINE NUMBER(10);
NO_OF_DAYS NUMBER(10);

BEGIN

t_rollno := :roll_no;
t_nameofbook := :nameofbook;

select DATEOFISSUE into ISSUEDATE from borrower where ROLL_NO = t_rollno and NAMEOFBOOK
LIKE '%' || t_nameofbook || '%';
```

```
DBMS_OUTPUT.PUT_LINE(' ISSUED ON ' || ISSUEDATE || CHR(10) || 'NO. OF. DAYS ' ||
ROUND(SYSDATE - ISSUEDATE));
NO_OF_DAYS := ROUND(SYSDATE- ISSUEDATE);
TOTAL_FINE := (CASE WHEN NO_OF_DAYS < 16 THEN 0
            WHEN NO_OF_DAYS < 30 THEN 5 * NO_OF_DAYS
            ELSE 50 * NO_OF_DAYS
            END);
IF TOTAL_FINE < 1 THEN
 DBMS_OUTPUT.PUT_LINE(CHR(10) || 'NO FINE' ||CHR(10));
ELSE
 DBMS_OUTPUT.PUT_LINE(CHR(10) || 'FINE:-> ' ||CHR(10) || TOTAL_FINE);
 INSERT INTO FINE(ROLL_NO, AMT) VALUES (t_rollno, TOTAL_FINE);
END IF;

UPDATE BORROWER
SET STATUS = 'R'
WHERE ROLL_NO = t_rollno and NAMEOFBOOK LIKE '%' || t_nameofbook || '%';
EXCEPTION
  WHEN no_data_found THEN
    dbms_output.put_line('No such Entry!');
  WHEN others THEN
    dbms_output.put_line('Error!');
END;
/
```

SQL IMPLICIT CURSOR
https://docs.oracle.com/cd/B28359_01/appdev.111/b28370/sql_cursor.htm#LNPLS01348

SQL CURSORS EXPLANATION
https://bhavanakhivsara.files.wordpress.com/2017/06/mysql-6.pdf

https://stackoverflow.com/questions/26041719/how-to-execute-a-stored-procedure-in-oracle-11g
HOW TO EXECUTE STORED PROCEDURE

PL/SQL Stored Procedure and Stored Function. Write a Stored Procedure namely proc_Grade for the categorization of student. If marks scored by students in examination is <=1500 and marks>=990 then student will be placed in distinction category if marks scored are between 989 and900 category is first class, if marks 899 and 825 category is Higher Second Class Write a PL/SQL block for using procedure created with above requirement. Stud_Marks(name, total_marks) Result(Roll,Name, Class)

```
create table Stud_Marks(ROLL_NO number(10), name varchar2(180), total_marks float);
```
⇨ Table created
```
create table Result(ROLL_NO NUMBER(10), NAME VARCHAR2(180), CLASS VARCHAR2(10));
```
⇨ Table created
```
DECLARE
    ROLL_NO1 STUD_MARKS.ROLL_NO%TYPE;
    NAME1 STUD_MARKS.NAME%TYPE;
```

```
    TOTAL_MARKS1 STUD_MARKS.TOTAL_MARKS%TYPE;
    CATEGORY RESULT.CLASS%TYPE;
 CURSOR C1
 IS
  SELECT ROLL_NO, NAME, TOTAL_MARKS FROM STUD_MARKS;

BEGIN
 OPEN C1;
 LOOP
 FETCH C1 INTO ROLL_NO1, NAME1, TOTAL_MARKS1;
  EXIT WHEN C1%NOTFOUND;
 CATEGORY := (CASE WHEN TOTAL_MARKS1>989 THEN 'DISTINCTION'
                   WHEN TOTAL_MARKS1>899 THEN 'FIRST CLASS'
                   WHEN TOTAL_MARKS1>824 THEN 'HIGHER SECOND CLASS'
                   ELSE 'FAIL'
                   END);
 INSERT INTO RESULT VALUES(ROLL_NO1, NAME1, CATEGORY);
  DBMS_OUTPUT.PUT_LINE(C1%ROWCOUNT);
 END LOOP;
 CLOSE C1;
END;
```

PROCEDURE:

```
CREATE OR REPLACE PROCEDURE prc_msg(ROLL_NO1 IN INTEGER)
IS
BEGIN
INSERT INTO RESULT VALUES(ROLL_NO1, (SELECT NAME FROM STUD_MARKS WHERE ROLL_NO =
ROLL_NO1), CASE WHEN (SELECT TOTAL_MARKS FROM STUD_MARKS WHERE ROLL_NO =
ROLL_NO1) >989 THEN 'DISTINCTION'  WHEN (SELECT TOTAL_MARKS FROM STUD_MARKS WHERE
ROLL_NO = ROLL_NO1) >899 THEN 'FIRST CLASS'
 WHEN (SELECT TOTAL_MARKS FROM STUD_MARKS WHERE ROLL_NO = ROLL_NO1) >824 THEN
'HIGHER SECOND CLASS'
ELSE 'FAIL'
END);
DBMS_OUTPUT.PUT_LINE(ROLL_NO1 || ':- ' );

END;
```

EXECUTION::

```
BEGIN
prc_msg(100010);
END;
```

SELECT RESULT.ROLL_NO, RESULT.NAME, RESULT.CLASS, STUD_MARKS.TOTAL_MARKS FROM RESULT, STUD_MARKS WHERE RESULT.ROLL_NO = STUD_MARKS.ROLL_NO;

ROLL_NO	NAME	CLASS	TOTAL_MARKS
100001	Shreeya Ajay Chavan	DISTINCTION	1200
100002	Sonal A. Chavan	FAIL	100
100003	Shraddha Bipen Mehta	FIRST CLASS	980
100004	Shruti Satish Salunke	DISTINCTION	1100
100005	Gauri Santosh Gad	HIGHER SECOND CLASS	825
100006	Shraddha Shailesh Bhosale	HIGHER SECOND CLASS	880
100007	Patricia Patrick D'gama	DISTINCTION	1500
100008	Amruta K Raut	DISTINCTION	1400
100009	Namrta C Pandey	FAIL	700
100010	Kirti R Maheshwr	FAIL	600

10 rows returned
in 0.00 seconds

```
   CREATE OR REPLACE PROCEDURE sum_salaries
AS
   p_sum INTEGER;
   p_sal INTEGER;
   CURSOR c IS SELECT TOTAL_MARKS FROM STUD_MARKS;
  BEGIN
    p_sum := 0;
    OPEN c;
   LOOP
    FETCH c INTO p_sal;
    EXIT WHEN c%NOTFOUND;
      p_sum := p_sum + p_sal;
      FETCH c INTO p_sal;
    END LOOP;
    CLOSE c;
    dbms_output.put_line('hii' || p_sum);
END;\
```

Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers). Write a database trigger on Library table. The System should keep track of the records that are being updated or deleted. The old value of updated or deleted records should be added in Library_Audit table.

```
CREATE OR REPLACE TRIGGER TRIG123 BEFORE UPDATE OR DELETE ON BORROWER
FOR EACH ROW
DECLARE
msal number(10);
BEGIN
```

```
INSERT INTO BORROWER2 VALUES (:old.ROLL_NO, :old.DATEOFISSUE, :old.NAMEOFBOOK,
:old.STATUS);
END;

update BORROWER
SET STATUS = 'I'
WHERE ROLL_NO = 100002;


delete from BORROWER
```

Mongo Db aggregation and indexing:
http://pradipshewale.blogspot.com/2015/09/aggregation-and-indexing-with-suitable.html

MongoDB CRUD Operations:
https://docs.mongodb.com/manual/crud/

Implement MapReduce

https://bhavanakhivsara.files.wordpress.com/2017/06/mongodb-map-reduce-examples-in-mongodb.pdf

JOINS

Natural Join

```
SELECT * FROM STUDENT NATURAL JOIN INSTRUCTOR
ORDER BY ROLL_NO, INST_ID;
--All rows matched with eachother
```

USING CLAUSE:

```
SELECT * FROM STUD_INST JOIN INSTRUCTOR USING(INST_ID)


SELECT * FROM STUD_INST JOIN INSTRUCTOR USING(INST_ID) WHERE INST_ID = 2


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

INNER JOIN

```
SELECT  STUD_INST.SID, FIRST_NAME, MIDDLE_NAME, LAST_NAME, PERCENTAGE,
STUD_INST.INST_ID, INST_FIRST_NAME, INST_LAST_NAME
FROM (STUDENT INNER JOIN STUD_INST ON (STUDENT.ROLL_NO = STUD_INST.SID)) INNER JOIN
INSTRUCTOR ON INSTRUCTOR.INST_ID = STUD_INST.INST_ID;
```

6 ROWS RETURNED

```
SELECT ROLL_NO, FIRST_NAME, MIDDLE_NAME, LAST_NAME, PERCENTAGE, INST_ID from
STUDENT LEFT JOIN STUD_INST ON STUDENT.ROLL_NO = STUD_INST.SID;
```

ROLL_NO	FIRST_NAME	MIDDLE_NAME	LAST_NAME	PERCENTAGE	INST_ID
100001	Shreeya	Ajay	Chavan	84	1
100002	Sonal	A.	Chavan	85	2
100003	Shraddha	Bipen	Mehta	86	3
100004	Shruti	Satish	Salunke	87	4
100005	Gauri	Santosh	Gad	88	5
100006	Shradha	Shailesh	Bhosale	89	6
100008	Amruta	K	Raut	81	-
100010	Kirti	R	Maheshwr	83	-
10015	SonNalii	ajaa12	chintooo	94	-
100007	Patricia	Patrick	D'gama	80	-
100009	Namrta	C	Pandey	82	-

11 rows returned in 0.00 seconds

```
SELECT ROLL_NO, FIRST_NAME, MIDDLE_NAME, LAST_NAME, PERCENTAGE, INST_ID from
STUDENT FULL OUTER JOIN STUD_INST ON STUDENT.ROLL_NO = STUD_INST.SID;
```

ROLL_NO	FIRST_NAME	MIDDLE_NAME	LAST_NAME	PERCENTAGE	INST_ID
100007	Patricia	Patrick	D'gama	80	-
100008	Amruta	K	Raut	81	-
100009	Namrta	C	Pandey	82	-
100010	Kirti	R	Maheshwr	83	-
100001	Shreeya	Ajay	Chavan	84	1
100002	Sonal	A.	Chavan	85	2
100003	Shraddha	Bipen	Mehta	86	3
100004	Shruti	Satish	Salunke	87	4
100005	Gauri	Santosh	Gad	88	5
100006	Shradha	Shailesh	Bhosale	89	6
10015	SonNalii	ajaa12	chintooo	94	-

11 rows returned in 0.07 seconds

SELECT ROLL_NO, FIRST_NAME, MIDDLE_NAME, LAST_NAME, PERCENTAGE, INST_ID from STUD_INST RIGHT JOIN STUDENT ON STUDENT.ROLL_NO = STUD_INST.SID;

11 rows returnd

CROSS JOIN

SELECT ROLL_NO, FIRST_NAME, INST_ID, INST_FIRST_NAME FROM STUDENT CROSS JOIN INSTRUCTOR

SUBQUERY

```
SELECT *
FROM employees
WHERE salary =
  (
    SELECT max(salary)
    FROM employees
  );
```

CURSOR

create table student(ID int,NAME varchar(50));

insert into student(ID,NAME) values(1,"indu Sharma");
insert into student(ID,NAME) values(2,'gulnarbanu’);
insert into student(ID,NAME) values(3,ashu');
insert into student(ID,NAME) values(4,'anu Wagh');
insert into student(ID,NAME) values(5,'pashya Patki');

create table student_load(ID int,NAME varchar(50));
insert into student_load(ID,NAME) values(6,"Suraj More ");
insert into student_load(ID,NAME) values(6,"Umran Shaikh");
insert into student_load(ID,NAME) values(7,'Sanket Choudhary');

insert into student load(ID,NAME) values(8,'Pallavi Thakre');

```
declare

cursor student is
select b.ID as rid ,a.ID,a.NAME from student a
left outer join student_load b
on a.ID=b.ID and a.NAME <> b.NAME;

type t__data is table of student%rowtype index by binary_integer;
t_data t__data;

begin
open student;
loop
 fetch student bulk collect into t_data limit 5;

 exit when t_data.count=0;

 for i in t_data.first..t_data.last loop
   if t_data(i).rid is null then
     insert into student_load(ID,NAME) values(t_data(i).ID,t_data(i).NAME);
   else
     update student_load
set NAME=t_data(i).NAME where ID=t_data(i).rid;
  end if;
 end loop;
end loop;
close student;
end;
```

FUNCTION

```
CREATE OR REPLACE FUNCTION FUNCURR( name_in IN varchar2 ) RETURN VARCHAR
AS

cursor student is
select b.ID as rid ,a.ID,a.NAME from student a
```

```
left outer join student_load b
on a.ID=b.ID and a.NAME <> b.NAME;

type t__data is table of student%rowtype index by binary_integer;
t_data t__data;

begin
open student;
loop
  fetch student bulk collect into t_data limit 5;

  exit when t_data.count=0;

  for i in t_data.first..t_data.last loop
    if t_data(i).rid is null then
      insert into student_load(ID,NAME) values(t_data(i).ID,t_data(i).NAME);
    else
      update student_load
set NAME=t_data(i).NAME where ID=t_data(i).rid;
   end if;
  end loop;
end loop;
close student;
DBMS_OUTPUT.PUT_LINE(name_in);
RETURN 'SUCCESS';
end;
```

Function Created

CREATE A FUNCTION FOR FACTORIAL

```
SELECT FUN23(5) FROM DUAL;

CREATE OR REPLACE FUNCTION FUN23(num1 IN number) RETURN INTEGER
AS
 num number;
 res int default 1;
BEGIN
  num := num1;
 while num >= 1 loop
  res := res * num;
  num := num - 1;
 END LOOP;
 RETURN res;
END;
```