Thadomal Shahani Engineering College

Bandra (W.), Mumbai - 400 050.

CERTIFICATE

Certify that ~~Mr/Miss~~ CHIRAG JAIN

of EXTC Department, Semester VI with

Roll No. 20 has completed a course of the necessary

experiments in the subject DBMS under my

supervision in the **Thadomal Shahani Engineering College**

Laboratory in the year **2019-2020**

Teacher In- Charge

Head of the Department

Date 28/06/2020

Principal

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

CONTENTS

EXPERIMENT-1

AIM - Create a database for library with necessary attributes and data types and implement DDL commands.

PROBLEM STATEMENT - Design and develop a database for the library for which details are as follows:-

a. To maintain information about all the students like student name, student ID, contact number, fine, address, email ID.
b. To maintain information about all the teachers like teacher like teacher name, teacher ID, contact number, fine, address, email ID.
c. To maintain information about books like book name, book number, Author name, Publication name.
d. To maintain information about staff working in library like Name, Staff ID, age, Contact, Email ID.
e. To maintain information about book taker by students like name of book, date of taking the book and date of returning.

TABLES DESIGNED –

Student (student_name, id, contact_no, fine, address, email_id)

Teachers (teachers_name, id, contact_no, address, email_id)

Books (book_name, book_ id, author_name, publication_name,)

Staff (staff_name, staff_id, staff_role, staff_role, age, contact_no, email_id)

THEORY – Database is collection which is related to data and database management in the system is a collection.

SQL – It is a programming language used in programming and designed for managing data.

DDL – The full form is Data Definition Language. It has a syntax similar to Computed Programming and designed for managing data.

VARIOUS DDL COMMANDS –

1. **CREATE** – It is used to create table in SQL by using the following syntax
   SYNTAX – Create table <table_name>

```
(
    Column_name datatype(size)
    .
);
```

   EXAMPLE – Create table student

```
(
    student_name (20),
    student_id char (20),
    contact_no numeric,
    fine int
);
```

2. **ALTER** - It is used to alter the contents of tables such as add, modify
   SYNTAX –Alter table <table_name> add column_name datatype ;
   Alter table <table_name> modify column_name datatype ;
   Alter table <table_name> drop column_name datatype ;

   EXAMPLE - Alter table student add address varchar (40);
   Alter table student modify address varchar(20);
   Alter table student contact_no drop numeric;

3. **TRUNCATE**- It is used to remove the contents of the table while the prototype of table remains.
   SYNTAX – Truncate table <table_name>;
   EXAMPLE – Truncate table student;

4. **DROP** – It is used to remove the entire table along with all its contents.
   SYNTAX –Drop table <table_name>;
   EXAMPLE –Drop table student;

5. **RENAME** – It is used to remove the table name and name it something else.
   SYNTAX –Rename table <table_name> to <table_name>;
   EXAMPLE –Drop table student to students_in_library;

6. **INSERT** – It is used to insert value in table.
   SYNTAX –Insert into <table_name> (col1, col2 ……, coln)
   Values (val1, val2, ……., valn)
   EXAMPLE – Insert into student (student_name, student_id, contact_no, fine)
   Values ('Riya', '1234', '98646759343', '0')

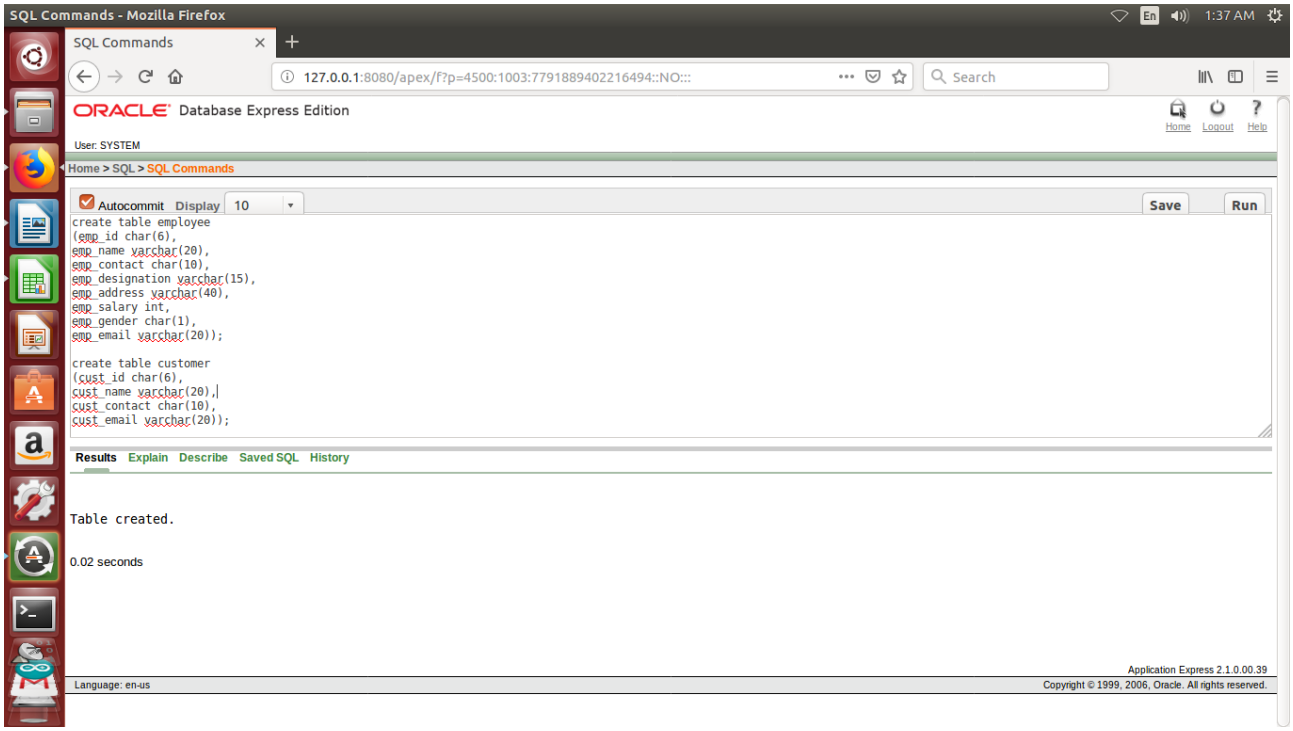
7. **DESCRIBE** – Some basic block of ER Diagram are entity-attribute relationship.
   SYNTAX – desc <table_name>
   EXAMPLE – desc student

CONCLUSION – A database of a library was created successfully using DDL commands.

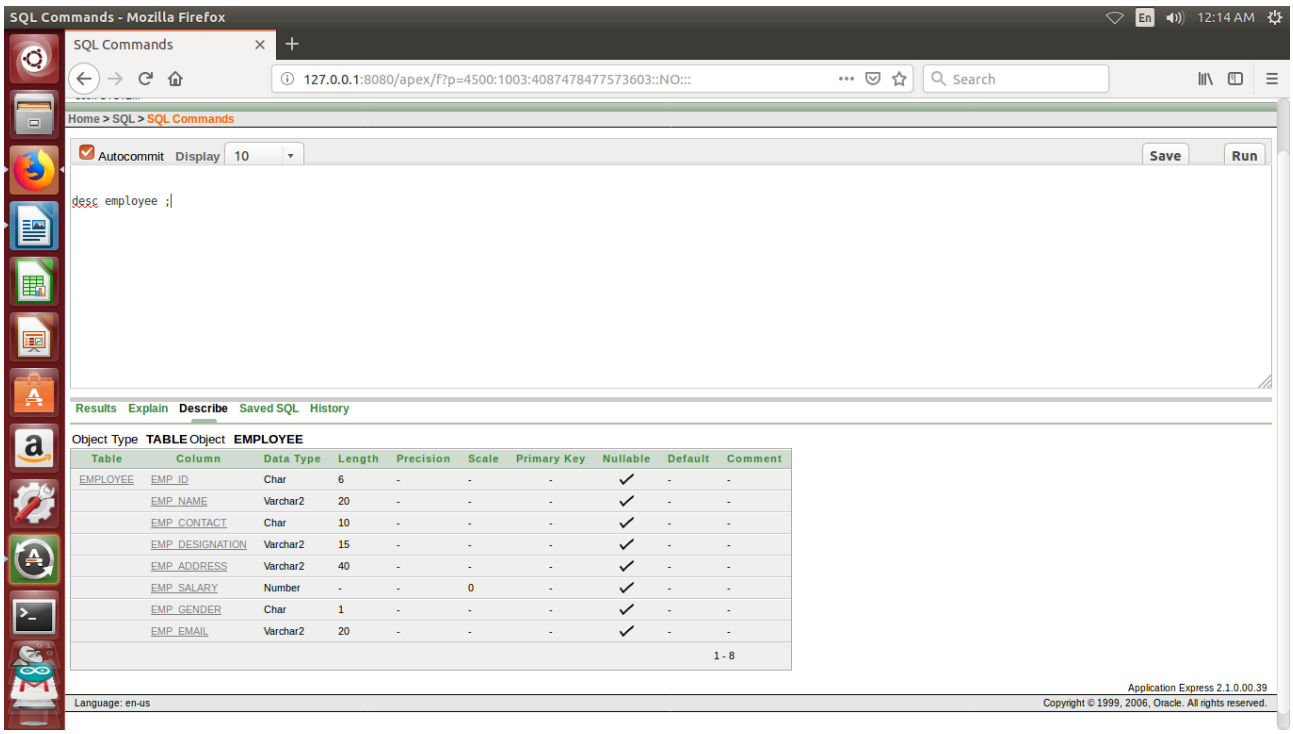
EXPERIMENT NO 1: CREATE A DATABASE FOR A GARMENT STORE

NAME: ANANYA **DATE:** 20/1/2020 **ROLL NO:** 4

QUERY: Create two tables, ‘employee’ and ‘customer’



QUERY: Describe the table ‘employee’



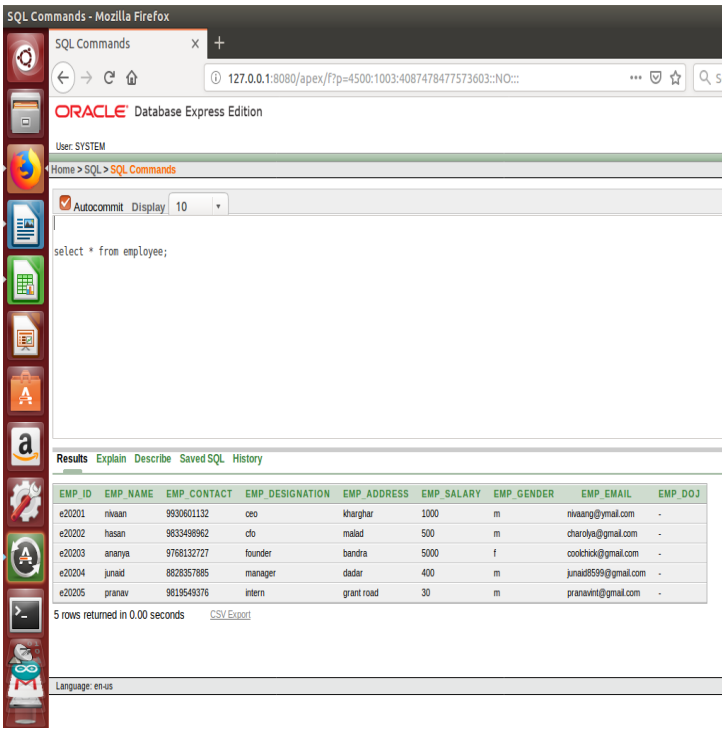
QUERY: Alter the table ‘employee’ by adding a new column

SQL Worksheet

```
1  alter table employee add emp_doj date;
```

```
Table altered.
```

ORACLE Integrated Cloud Applications & Platform Services



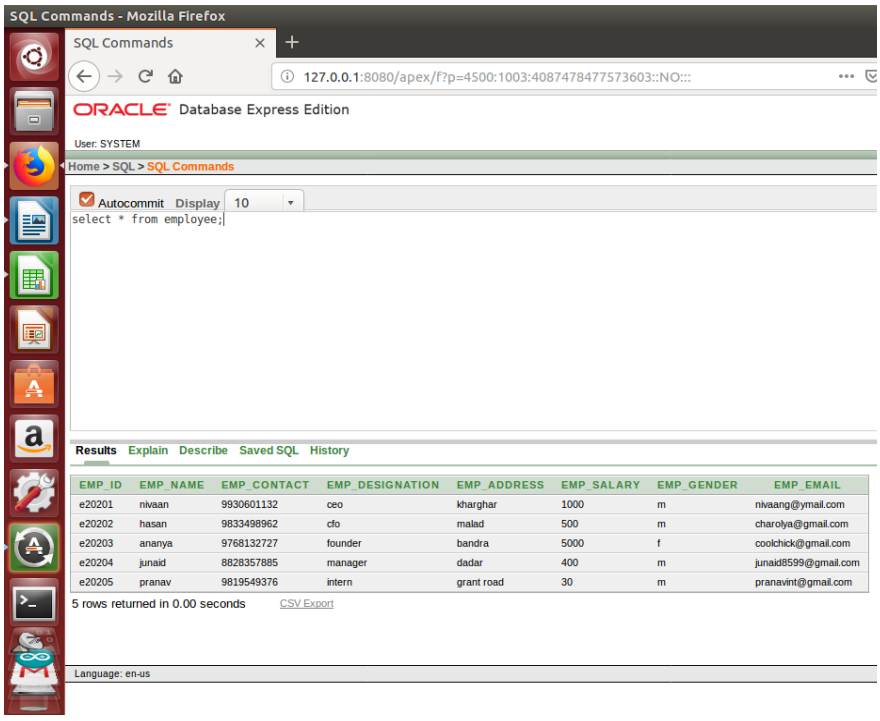
QUERY: Alter the table ‘employee’ by dropping a column

SQL Worksheet

```
1  alter table employee drop column emp_doj ;
```

```
Table altered.
```

ORACLE Integrated Cloud Applications & Platform Services



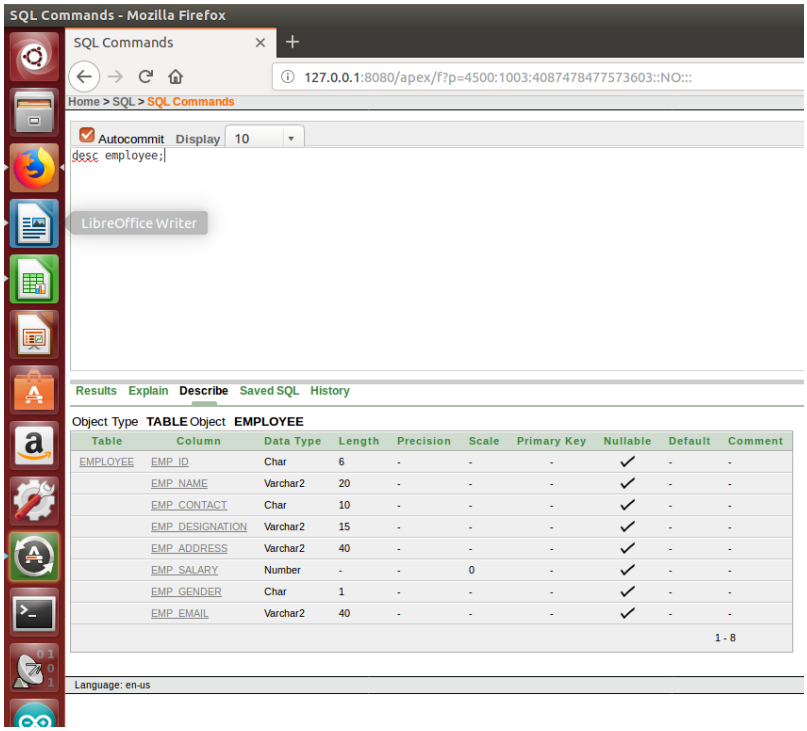
QUERY: Alter the table ‘employee’ and modify the datatype of the email attribute

SQL Worksheet

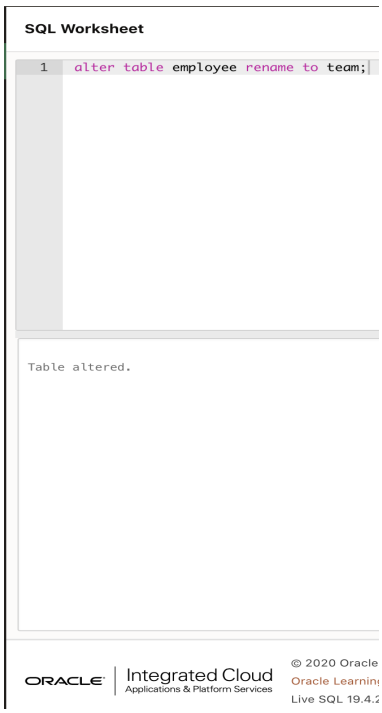
```
1  alter table employee modify emp_email varchar(40);
```

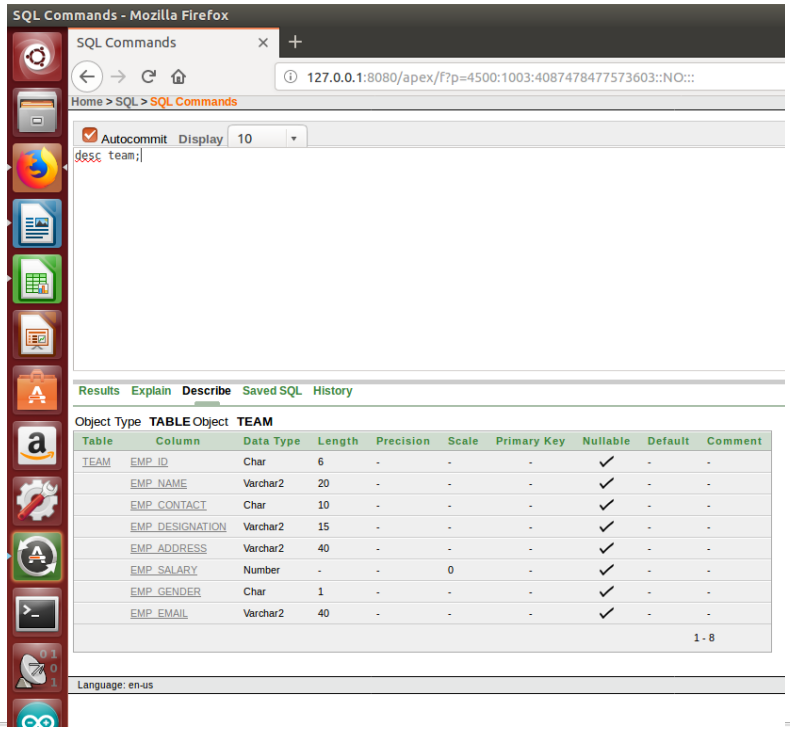
```
Table altered.
```

ORACLE | Integrated Cloud
Applications & Platform Services

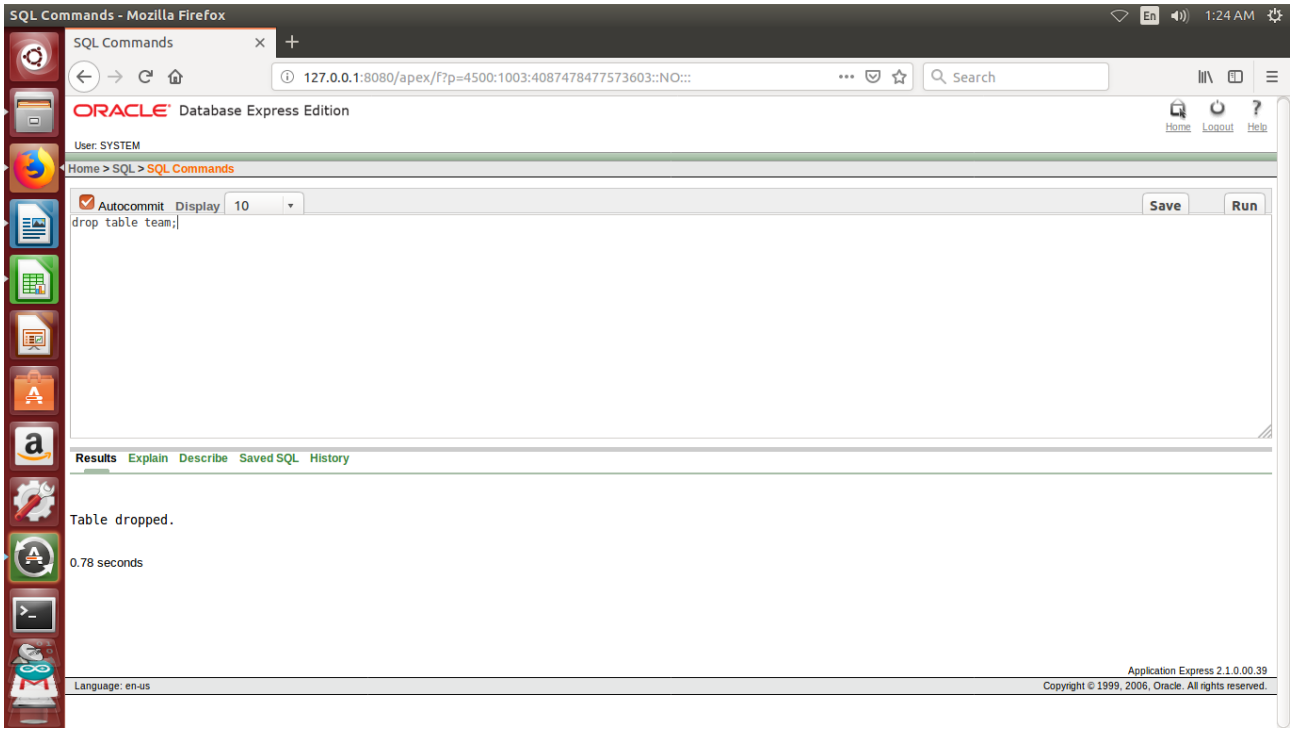


QUERY: Alter the table ‘employee’ by renaming it to ‘team’

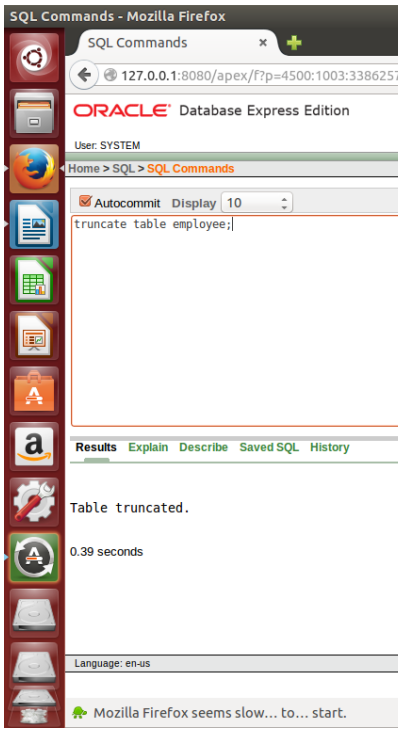


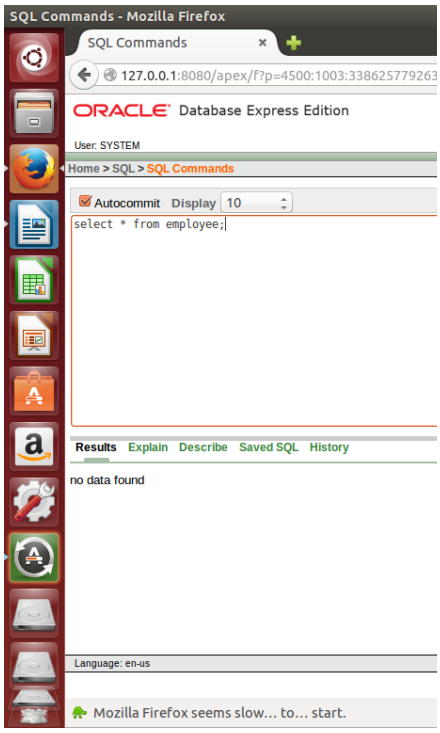


QUERY: Drop the table ‘team’



QUERY: Truncate the table ‘employee’


SQL Commands - Mozilla Firefox
SQL Commands
127.0.0.1:8080/apex/f?p=4500:1003:3386257
ORACLE Database Express Edition
User: SYSTEM
Home > SQL > SQL Commands
Autocommit Display 10
truncate table employee;
Results Explain Describe Saved SQL History
Table truncated.
0.39 seconds
Language: en-us
Mozilla Firefox seems slow... to... start.


SQL Commands - Mozilla Firefox
SQL Commands
127.0.0.1:8080/apex/f?p=4500:1003:338625779263
ORACLE Database Express Edition
User: SYSTEM
Home > SQL > SQL Commands
Autocommit Display 10
select * from employee;
Results Explain Describe Saved SQL History
no data found
Language: en-us
Mozilla Firefox seems slow... to... start.

EXPERIMENT NO. 02

IMPLEMENTING DML COMMANDS LIKE INSERT, UPDATE AND DELETE

Aim: To write SQL statements for implementing data manipulation language (update, insert, delete).

Theory: Data Manipulation Language

1. Definition- It is a family of computer languages including comments permitting users to manipulate data in database. This manipulation involves inserting data into the database tables, reviewing existing data, deleting data from already existing database and also modifying the existing data from tables.

2. Following are some SQL DML commands-
   i. INSERT: It is used to add a single tuple to a relation, and to add records into an existing table.
   Syntax:
   - To insert values in random order –
     INSERT into table_name(col1, col3)
     values(vale1, value3);

   - To insert values in order –
     INSERT into table_name values(col1_val, col2_val,…..);

   - To insert values from another table –
     INSERT into table2_name(col1_val, ….)
     SELECT col1, ……,coln from table1_name;

   ii. DELETE: It is used to delete same or all records from the existing table.

   Syntax:

   - To delete all the records from a table –
     DELETE from table_name;
     Eg: DELETE from student;

- To delete specific records from a table:

  DELETE from table_name where(condition);

  Eg: DELETE from student where std_bkqty=0;

iii. UPDATE: It is used to modify the existing data present in a table. We can also use 'WHERE' clause with it.

Syntax:

- To update all the rows:

  UPDATE table_name set col_1=new_val;

  Eg: UPDATE student set std_gender='M';

- To update specific rows:

  UPDATE table_name set col_1=new_val WHERE(condn);

  Eg: UPDATE student set std_branch='EXTC'

  WHERE std_name='SAUD';

iv. SELECT: It is used to retrieve data from the table.

Syntax:

- To show all records-

  Select * from table_name

- To show certain records-

  Select col1, col2 from table_name;

- To display conditional records-

  Select * from table_name WHERE(condn);

- To replicate a table-

  Create table table_name as Select * from table_name;

- SELECT command syntax:

  SELECT * or {[distinct] col_list [an alias]}
  FROM table
  [WHERE clause]
  [GROUPBY clause]
  [HAVING clause]
  [ORDERBY clause]

  Eg: SELECT * from student;
  SELECT name, branch from student;
  SELECT * from student WHERE std_bkqty>4;
  SELECT table student_new as SELECT * from student;
  SELECT std_branch count(*) from student groupby std_branch;
  SELECT * from student orderby std_bkqty desc;

CONCLUSION:

DML commands were understood along with their implementation of syntax, cited examples, and the collection of these helped in the successful execution of the experiment.

EXPERIMENT NO 2: IMPLEMENTATION OF DML COMMANDS

NAME: ANANYA **DATE:** 20/1/2020 **ROLL NO:** 4

QUERY: Insert five records into table ‘employee’

SQL Worksheet

Clear

```
1  insert into employee values ('e20201', 'nivaan' , '9930601132' , 'ceo' , 'kharghar' , 1000 , 'm' , 'nivaang@ymail.com' );
2  insert into employee values ('e20202','hasan', '9833498962', 'cfo', 'malad', 500, 'm', 'charolya@gmail.com' );
3  insert into employee values ('e20203','ananya','9768132727','founder','bandra',5000,'f','coolchick@gmail.com');
4  insert into employee values('e20204','junaid','8828357885','manager','dadar',400,'m','junaid8599@gmail.com');
5  insert into employee values('e20205','pranav','9819549376','intern','grant road',30,'m','pranavint@gmail.com');
```

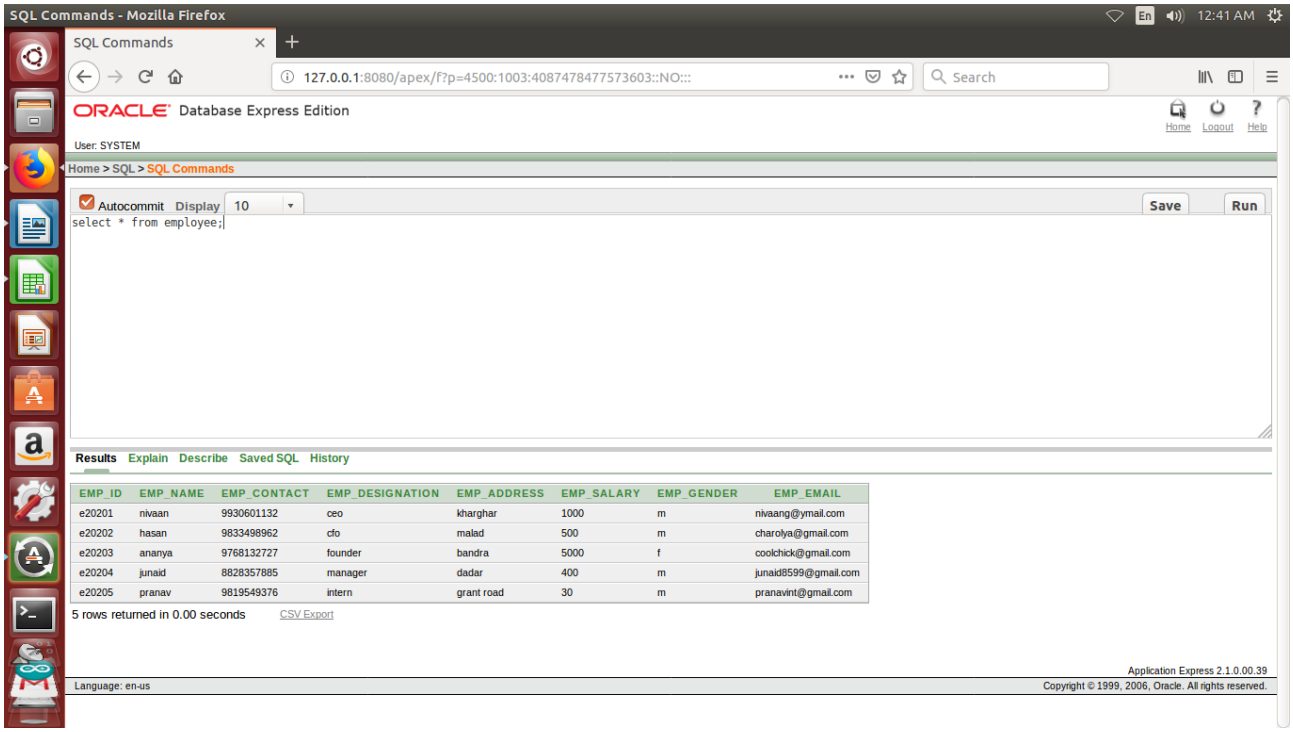
```
1 row(s) inserted.

1 row(s) inserted.
```

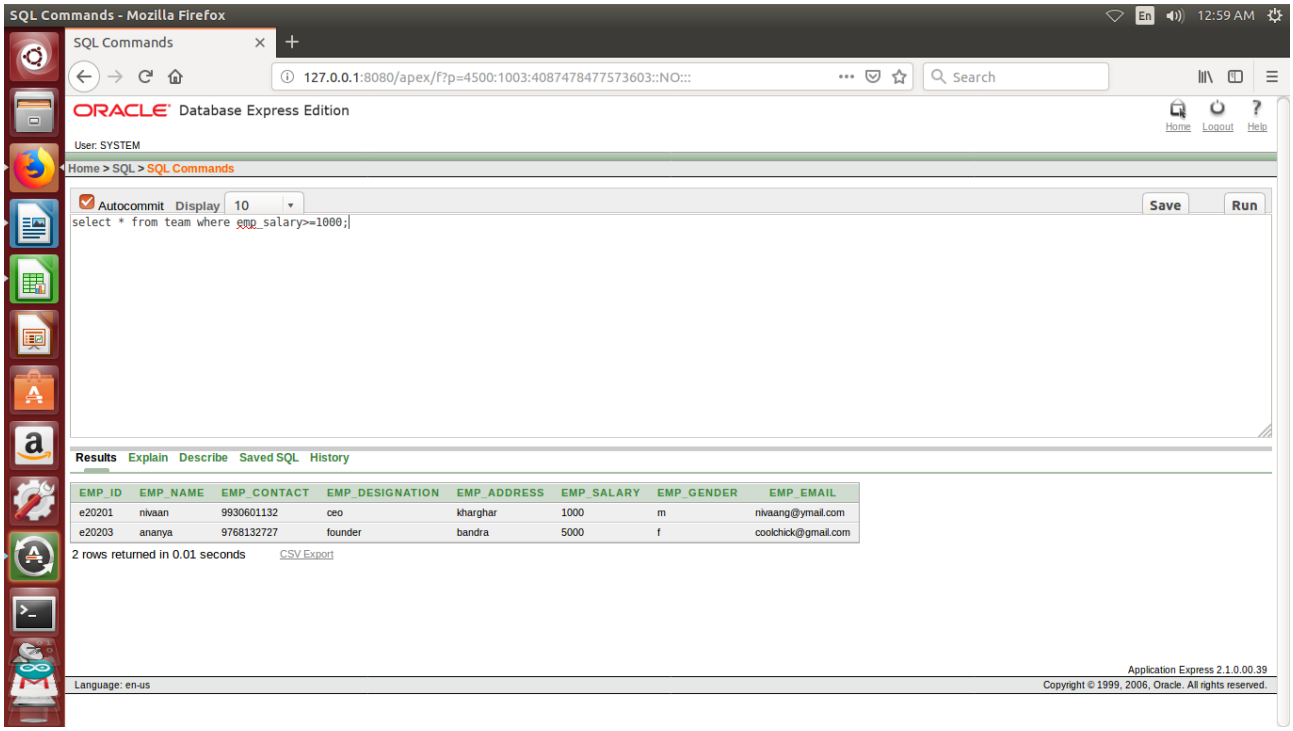
ORACLE | Integrated Cloud Applications & Platform Services

Oracle Learning Library · Ask Tom · Dev Gym · Database Doc 19c , 18c , 12c · Follow on Twitter

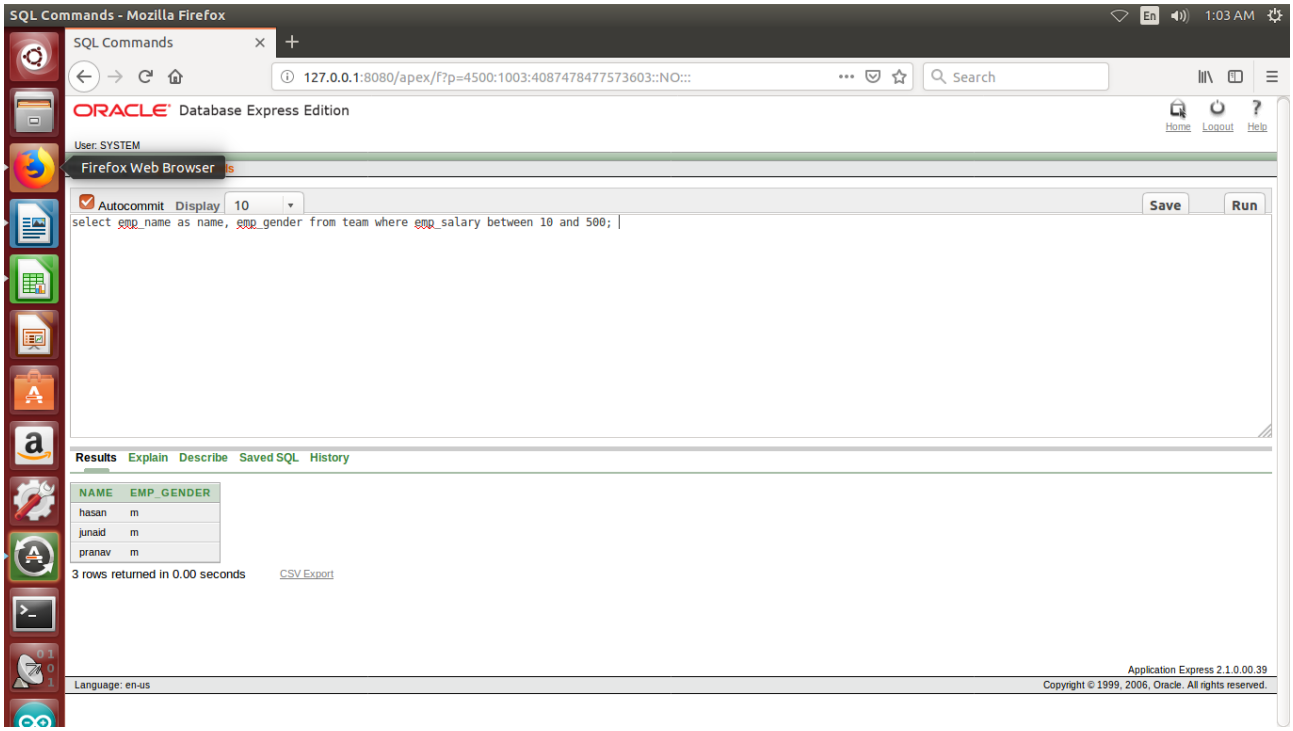
Live SQL 19.4.2, running Oracle Database 19c Enterprise Edition - 19.5.0.0.0 Built with ❤ using Oracle APEX



QUERY: Display records of employees with salary more than Rs 1000

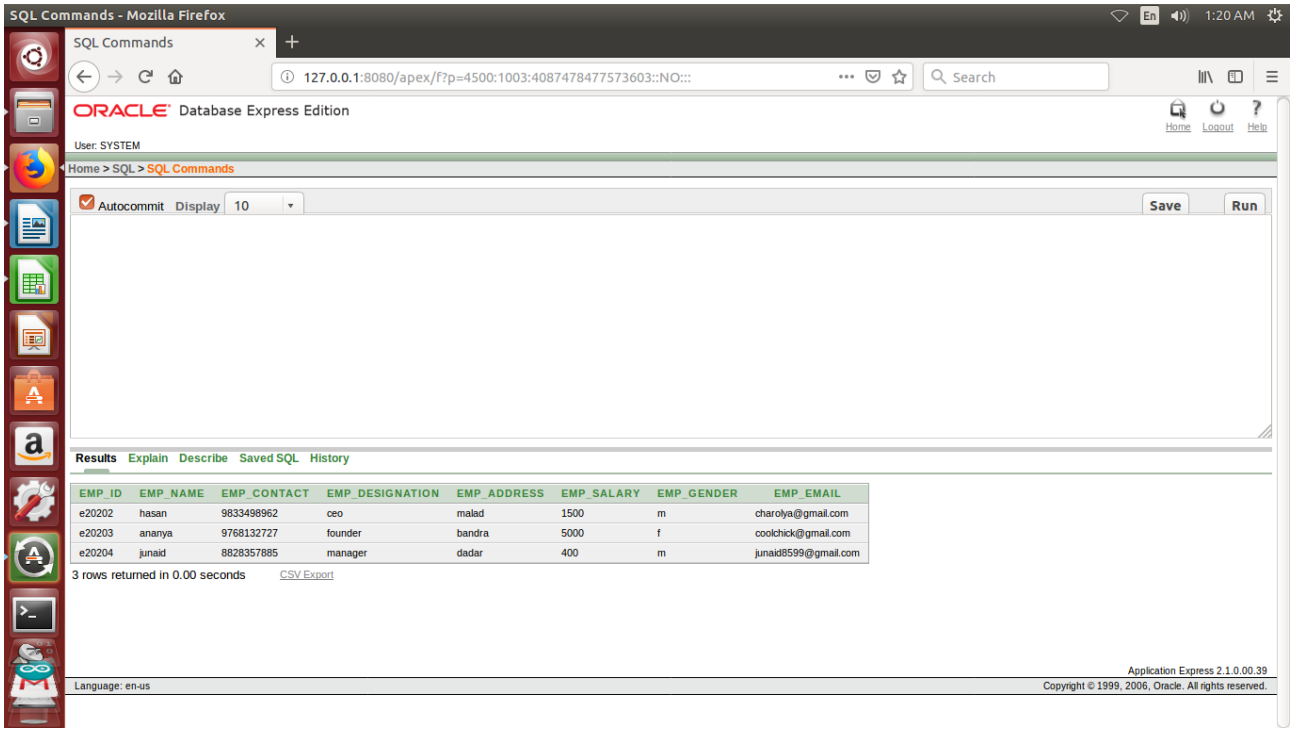


QUERY: Display the names and gender of employees with salary between Rs 10 and 500

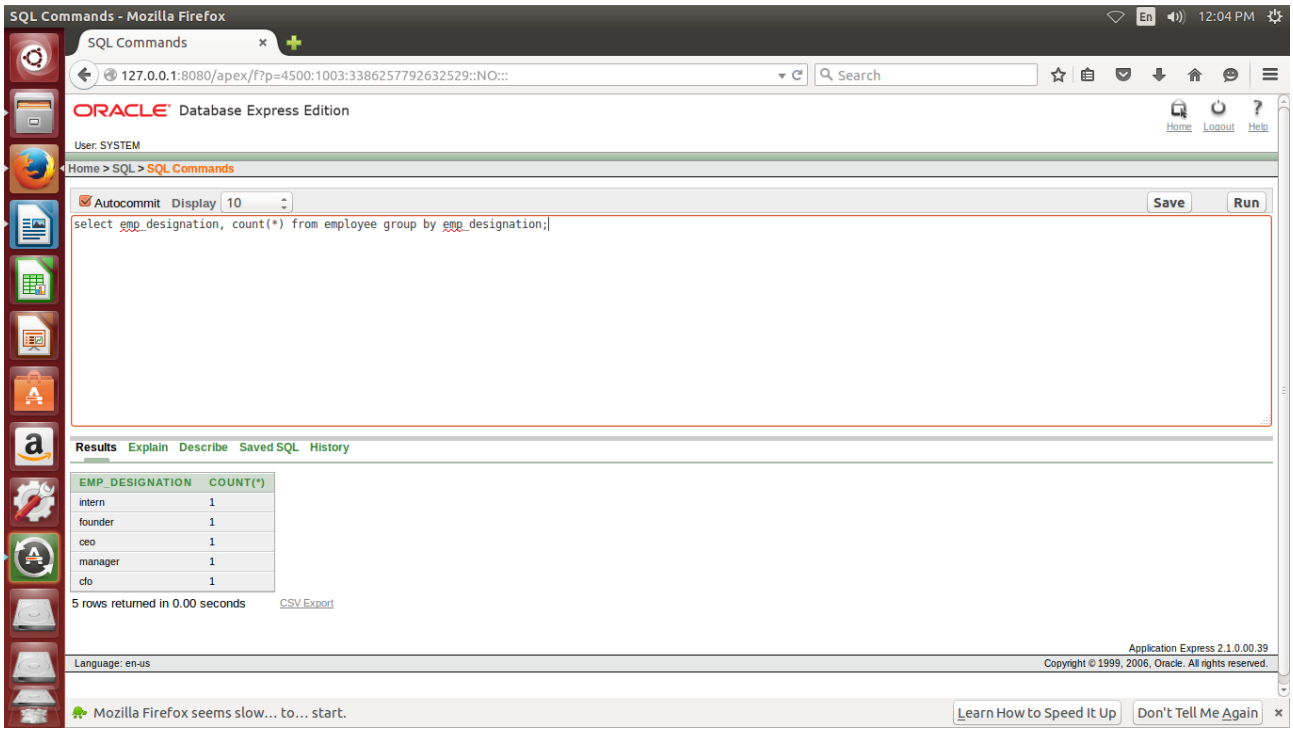


QUERY: Display records of employees staying in bandra, malad and dadar

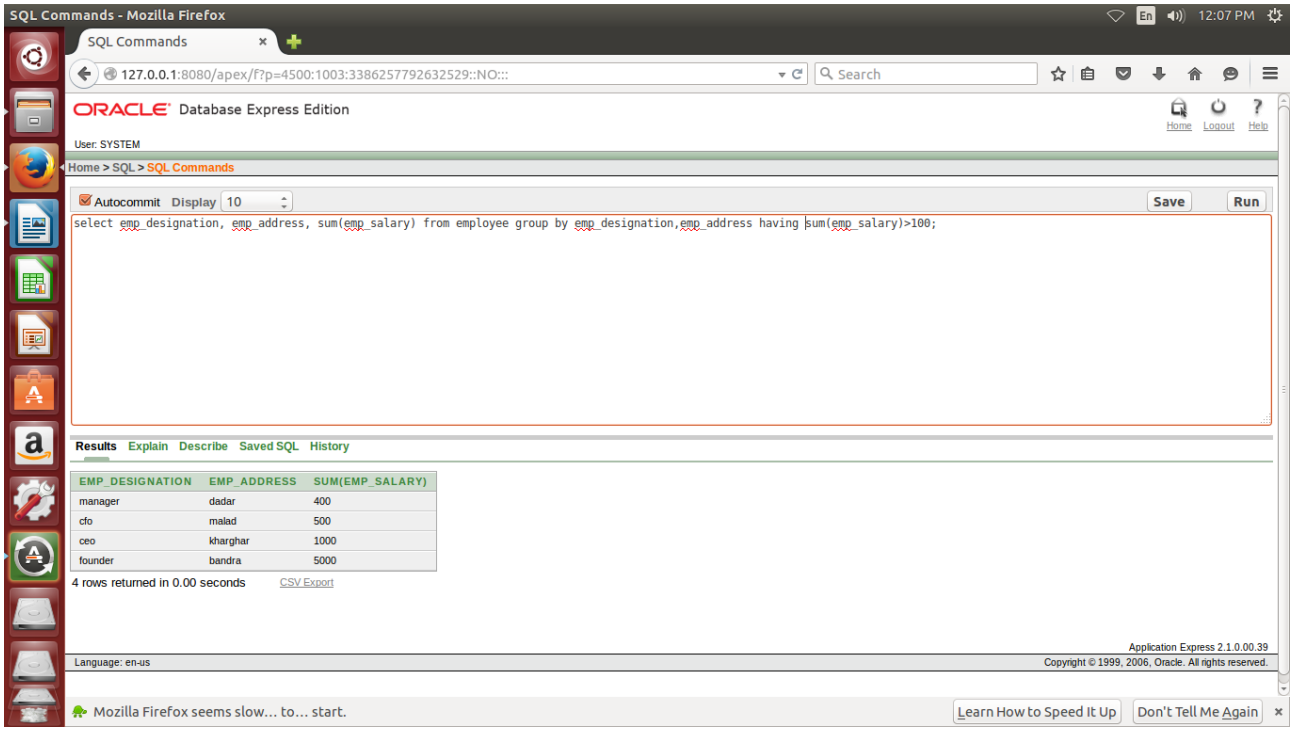
select * from team where emp_address in('bandra','malad','dadar');



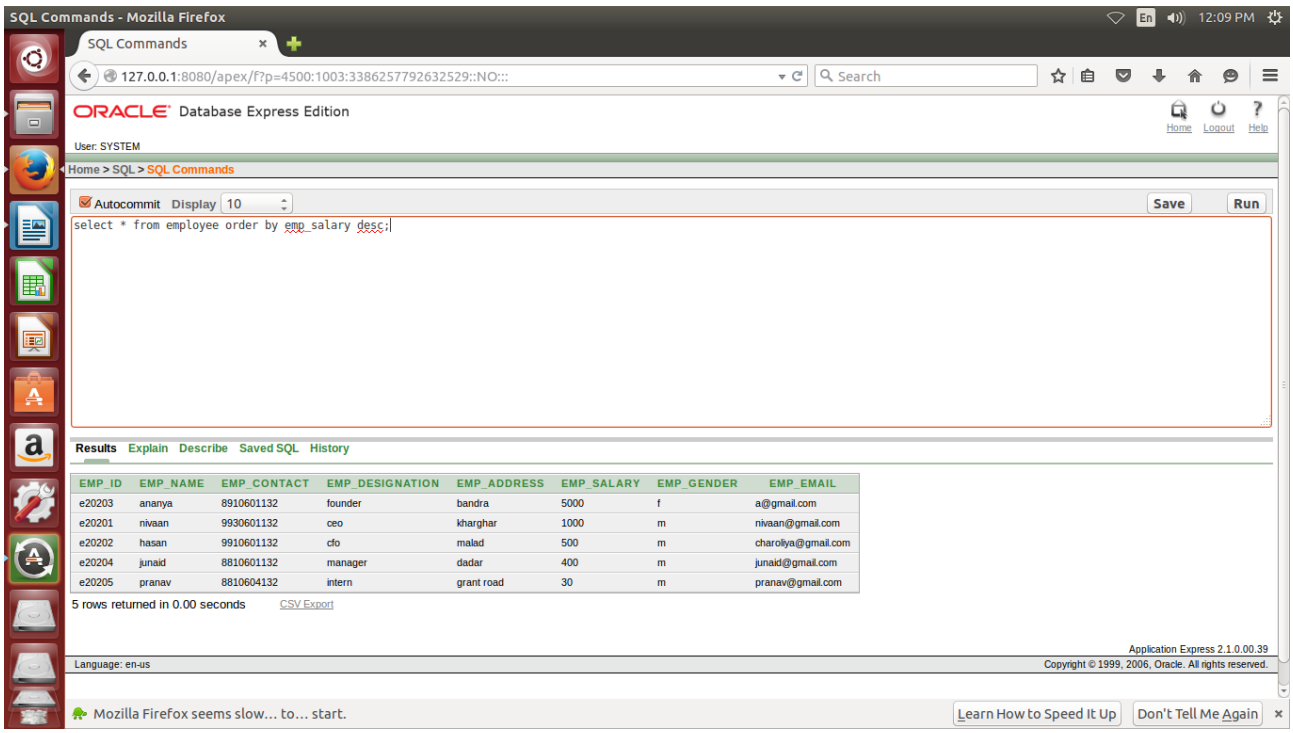
QUERY: Display count of employees grouped by ‘designation’



QUERY: Display and group employees with a salary greater than Rs 100 by ‘designation’ and ‘address’



QUERY: Display employees in descending order of their ‘salary’

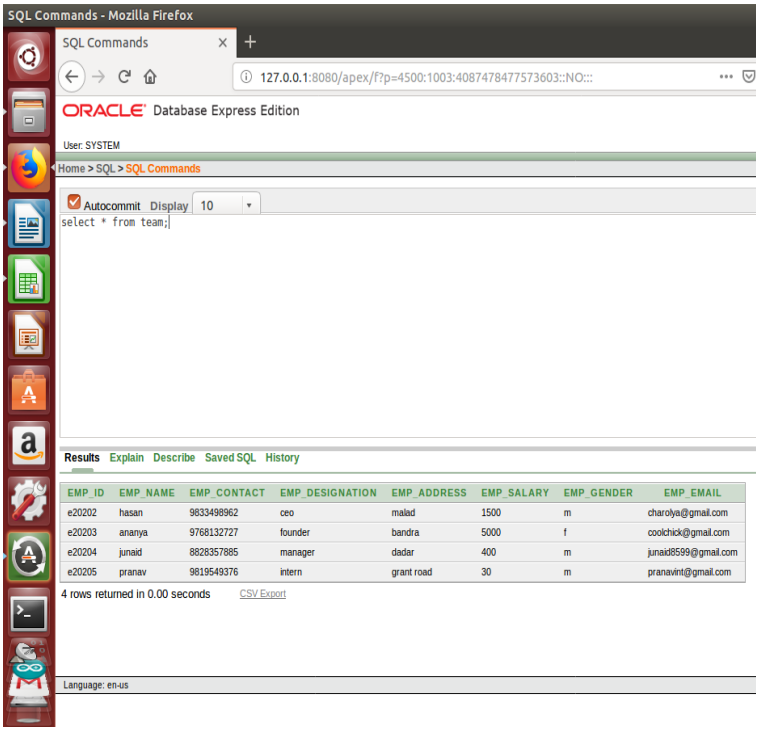


QUERY: Update designation and salary of employees with name ‘hasan’

SQL Worksheet

```
1  update team set emp_designation='ceo',emp_salary=emp_salary+1000 where emp_name='hasan';
2
3
```

```
1 row(s) updated.
```

QUERY: Delete from table, the record of employees whose address starts with letter ‘k’

SQL Worksheet

```
1  delete from team where emp_address like 'k%';
```

```
1 row(s) deleted.
```

ORACLE | Integrated Cloud
Applications & Platform Services

SQL Commands - Mozilla Firefox

SQL Commands

127.0.0.1:8080/apex/f?p=4500:1003:4087478477573603::NO:::

ORACLE Database Express Edition

User: SYSTEM

Home > SQL > SQL Commands

Autocommit Display 10

```
select * from team;
```

Results Explain Describe Saved SQL History

EMP_ID	EMP_NAME	EMP_CONTACT	EMP_DESIGNATION	EMP_ADDRESS	EMP_SALARY	EMP_GENDER	EMP_EMAIL
e20202	hasan	9833498962	cfo	malad	500	m	charolya@gmail.com
e20203	ananya	9768132727	founder	bandra	5000	f	coolchick@gmail.com
e20204	junaid	8828357885	manager	dadar	400	m	junaid8599@gmail.com
e20205	pranav	9819549376	intern	grant road	30	m	pranavint@gmail.com

4 rows returned in 0.00 seconds CSV Export

Language: en-us

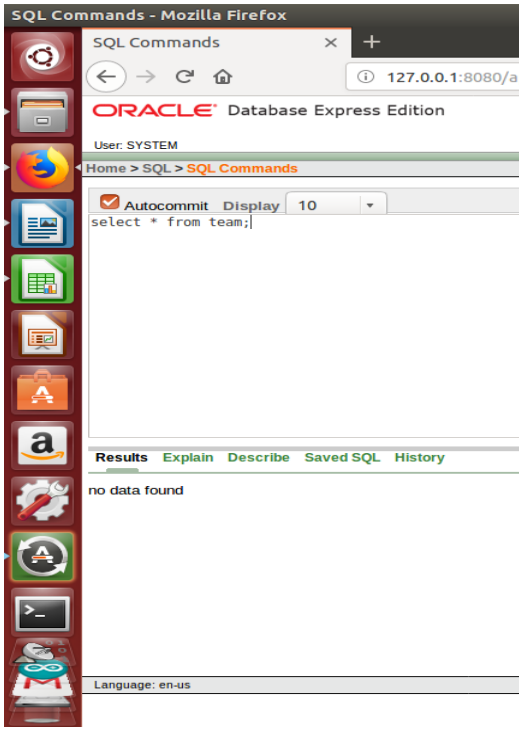
QUERY: Delete all data from the table ‘team’

SQL Worksheet

```
1  delete from team;
```

```
4 row(s) deleted.
```

ORACLE | Integrated Cloud
Applications & Platform Services



EXPERIMENT N0.- 3

AIM : To apply constraints like primary key , foreign key , not null , unique , check , default etc on designed database .

THEORY : Domain constraint

Every column in a database has a domain , a set of data values that are legal for that column . this can be achieved with the help of check constraint or table check constraint . A table check constraint is a condition that defines valid data when adding or updating , specifying check constraints is done through a restricted form of a search condition .

For example – in a table containing product one can check that the price of a product and quality of a product is a non – negative value i.e. price >0 , quality > 0

Check constraint are used to ensure the validity of data and to provide data integrity .

Eg : sql > create table t1

```
(
    Sno number ,
    Address varchar (20) constraints ch add check (address like “m%”)
);
```

Types of constraint :

1) Nullness constraints
2) Unique constraints
3) Primary key constraints
4) Foreign key constraints

1) **Nullness constraints :**
   Some attributes in database are not allowed to contain NULL values . Null values are values which are unknown , unassigned or missing values
   a) Null constraints:
      Some attributes in table are not data so such columns can be defined as null column . By default when we create table all attribute in table are null attributes i.e. null values are allowed in that field .

   b) Not null constraints :
      For certain attributes null values are not allowed in such case the can restrict ; null entities with help of not null constraints.
      Eg- create table students (s_id int NOT NULL , Name Varchar (60), Age int);
      In s_id field will not take not null value .

2) **Uniques constraints**
   In case of unique constraints no two tupples can have equal values for some attributes .
   This constraints says that attributes from candidate keys which allows and null values which is unique by itself , tis unique constraints can be applicable to user defined domain declaration also ,
   Eg- Create table students (s_id int NOT NULL UNIQUE , Name Varchar (60), Age int);

3) Primary key constraints

Primary key attributes in same a unique key constraints with NOT null constraints . it needs to be unique as well as null values are not allowed in primary key attributes . The main difference between unique and primary key is that null values is allowed in unique constraints and write null value is not allowed in primary key constraints .

Primary key = unique + not null

Eg - create table customer
(name char (25) not null
Id char (10)
Primary key (id)) ;

4) Foreign key constraints

A value appearing in a one relation (table) for a given set of attributes alos appear for another set of attributes in a another relatives . This is called Referential integrity .

Eg – for parent table

```
Sql > create table t1
        (Sno number primary key ,
         Sname varchar (20)
        );

        For child table

Sql > create table t2
        (Sno number reference t1(sno) ,
         subject varchar (20)
        );
```

CONCLUSION: constraints like primary key , foreign key , not null etc are implemented .

EXPERIMENT NO 3: APPLY CONSTRAINTS

NAME: ANANYA **DATE:** 3/2/2020 **ROLL NO:** 4

QUERY: Apply constraints not null & primary key while creating a new table ‘designers’

SQL Worksheet — Clear | Find | Actions | Save | Run

```
1  create table designers
2  (
3  des_id char(10) primary key,
4  des_name varchar(16) not null,
5  des_doj date,
6  des_contact char(10)
7  );
8
```

```
Table created.
```

Columns

#	Column	Type	Length	Precision	Scale	Nullable	Semantics	Comment
1	DES_ID	CHAR	10			No	Byte	
2	DES_NAME	VARCHAR2	16			No	Byte	
3	DES_DOJ	DATE	7			Yes		
4	DES_CONTACT	CHAR	10			Yes	Byte	

QUERY: Add a new column in ‘designers’ with constraint not null

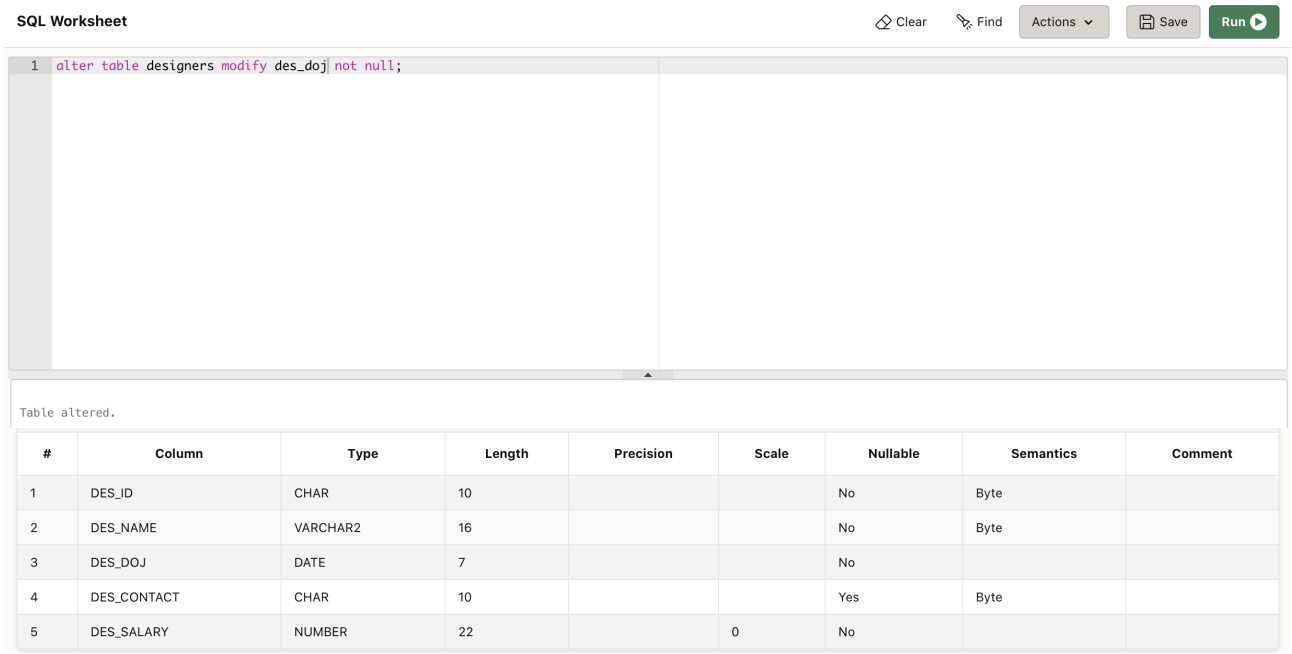
SQL Worksheet — Clear | Find | Actions | Save | Run

```
1  alter table designers add des_salary int not null;
```

```
Table altered.
```

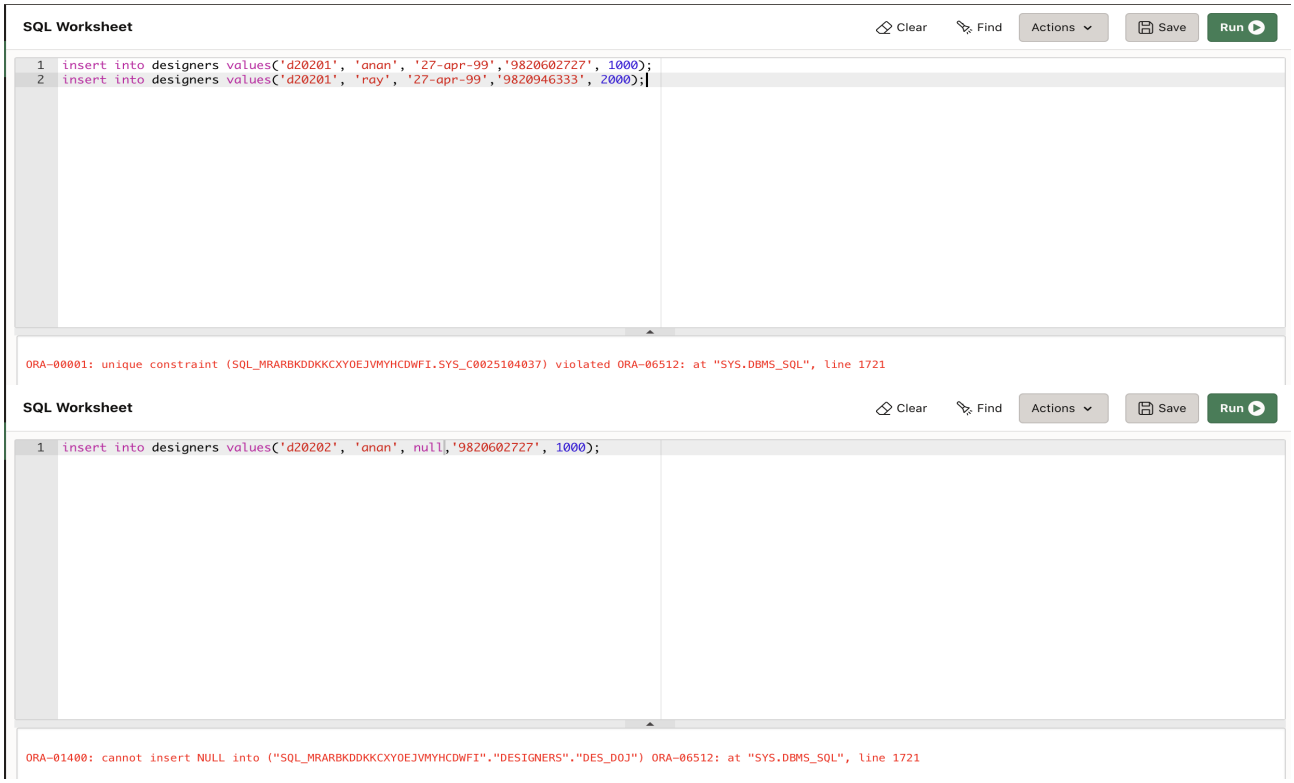
#	Column	Type	Length	Precision	Scale	Nullable	Semantics	Comment
1	DES_ID	CHAR	10			No	Byte	
2	DES_NAME	VARCHAR2	16			No	Byte	
3	DES_DOJ	DATE	7			Yes		
4	DES_CONTACT	CHAR	10			Yes	Byte	
5	DES_SALARY	NUMBER	22		0	No		

QUERY: Alter the table ‘designers’ and add a constraint to an existing column



#	Column	Type	Length	Precision	Scale	Nullable	Semantics	Comment
1	DES_ID	CHAR	10			No	Byte	
2	DES_NAME	VARCHAR2	16			No	Byte	
3	DES_DOJ	DATE	7			No		
4	DES_CONTACT	CHAR	10			Yes	Byte	
5	DES_SALARY	NUMBER	22		0	No		

QUERY: Insert a null value into a not null & non unique value in primary key column



QUERY: Create new table ‘material’ & add unique and check constraint

SQL Worksheet

Clear Find Actions Save Run

```
1  create table material
2  (
3  mat_quality varchar(10),
4  mat_length int,
5  mat_name varchar(20) unique,
6  check(mat_length>20)
7  );
```

```
Table created.
```

Columns

#	Column	Type	Length	Precision	Scale	Nullable	Semantics	Comment
1	MAT_QUALITY	VARCHAR2	10			Yes	Byte	
2	MAT_LENGTH	NUMBER	22		0	Yes		
3	MAT_NAME	VARCHAR2	20			Yes	Byte	

QUERY: Insert non unique value & an illegal value in a unique column & check column

SQL Worksheet

Clear Find Actions Save Run

```
1  insert into material values('good', 22, 'silk');
2  insert into material values('good', 25, 'silk');
```

```
ORA-00001: unique constraint (SQL_MRARBKDDKKCXYOEJVMYHCDWFI.SYS_C0025104968) violated ORA-06512: at "SYS.DBMS_SQL", line 1721
```

SQL Worksheet

Clear Find Actions Save Run

```
1  insert into material values('good', 10, 'silk');
```

```
ORA-02290: check constraint (SQL_MRARBKDDKKCXYOEJVMYHCDWFI.SYS_C0025104967) violated ORA-06512: at "SYS.DBMS_SQL", line 1721
```

QUERY: Add a column in table ‘designers’ with a default constraint

SQL Worksheet

Clear Find Actions Save Run

```
1  alter table designers add des_age int default 18;
2
```

Table altered.

#	Column	Type	Length	Precision	Scale	Nullable	Semantics	Comment
1	DES_ID	CHAR	10			No	Byte	
2	DES_NAME	VARCHAR2	16			No	Byte	
3	DES_DOJ	DATE	7			No		
4	DES_CONTACT	CHAR	10			Yes	Byte	
5	DES_SALARY	NUMBER	22		0	No		
6	DES_AGE	NUMBER	22		0	Yes		

QUERY: Insert into table ‘designers’ with no value in age

SQL Worksheet

Clear Find Actions Save Run

```
1  insert into designers values('d20202', 'ray','27-apr-99', '9820946333', 1000, default);
```

1 row(s) inserted.

SQL Worksheet

Clear Find Actions Save Run

```
1  select * from designers;
```

DES_ID	DES_NAME	DES_DOJ	DES_CONTACT	DES_SALARY	DES_AGE
d20201	anan	27-APR-99	9820602727	1000	18
d20202	ray	27-APR-99	9820946333	1000	18

Download CSV

2 rows selected.

QUERY: Drop unique constraint in table ‘material’

SQL Worksheet

```
alter table material drop unique(mat_name);
```

Table altered.

SQL Worksheet

```
insert into material values('good', 25, 'silk')
select * from material;
```

MAT_QUALITY	MAT_LENGTH	MAT_NAME
good	22	silk
good	25	silk

Download CSV
2 rows selected.

QUERY: Drop primary key constraint in table ‘designers’

SQL Worksheet

```
alter table designers drop primary key;
```

Table altered.

SQL Worksheet

```
insert into designers values('d20202','hh','21-may-99', '9820942216', 1000, 20);
select * from designers;
```

DES_ID	DES_NAME	DES_DOJ	DES_CONTACT	DES_SALARY	DES_AGE
d20201	anan	27-APR-99	9820602727	1000	18
d20202	hh	21-MAY-99	9820942216	1000	20
d20202	ray	27-APR-99	9820946333	1000	18

Download CSV
3 rows selected.

QUERY: Add foreign key constraint while creating table

SQL Worksheet

Clear | Find | Actions | Save | Run

```
1  create table complaints
2  (
3  complaint_id char(6) primary key,
4  complaint_desc varchar(40),
5  comp_id char(6) REFERENCES customer (cust_id)
6  )
```

```
Table created.
```

Schema \ **COMPLAINTS**

Syntax Help

Show All | Table Attributes | Columns | Indexes | Triggers | Constraints

#	Column	Type	Length	Precision	Scale	Nullable	Semantics	Comment
1	COMPLAINT_ID	CHAR	6			No	Byte	
2	COMPLAINT_DESC	VARCHAR2	40			Yes	Byte	
3	COMP_ID	CHAR	6			Yes	Byte	

Indexes

Index Name	Index Type	Uniqueness	Status	Columns
SYS_C0025111713	NORMAL	UNIQUE	VALID	COMPLAINT_ID

Triggers

No triggers defined.

Constraints

Constraint	Type	Condition	On Delete	Status	Last Change	Invalid?
SYS_C0025111714	Foreign Key	-	NO ACTION	ENABLED	17 seconds ago	-
SYS_C0025111713	Primary Key	-	-	ENABLED	17 seconds ago	-

QUERY: Add foreign key constraint while altering the table

SQL Worksheet

Clear | Find

```
1  alter table designers
2  add constraint fk_designers foreign key(emp_id)
3  references employee (emp_id)
```

```
Table altered.
```

EXPERIMENT-04

Aim: Write the queries to implement all join commands.

Theory: A join clause is used to combine rows from two or more tables, based on a related column between them. The relational operations can merge columns from two different tables to form a new Join Table. We can join multiple tables with the help of the same join condition (Equality or Inequality) or without any join condition (Cross Join).

Joining multiple tables using the Join condition. This type of join is useful when we know the joining condition.

Syntax: Select column list

From Table1, Table2

Where Join-condition

Based on joining condition Join can be divided into two types:

1) Equi-Join (Equal Join)
2) Non-Equi Join

1) Equi-Join (Equal Join)
The join query in which equality condition used for joining multiple tables is called equal joins.

2) Non Equi – Join
The join query in which equality condition is not used then such joins are called as non-equal joins.

Types of Join:

a) Natural Join
b) Cross Join (Cartesian)
c) Self Join
d) Outer Join
e) Inner Join

a) Natural Join
A Natural join returns all rows by matching values in common columns having the same name and data type of columns and that column should be present in both tables.
Natural join eliminates duplicate columns present in the Join table. Therefore common column will be printed only once in the resultant table.
Syntax: Select(column_name) from <Table1_name> natural join <Table2_name>
Eg: Select * from doc1 natural join doc2

b) Cross Join

The Cross join produces a result set which is the number of rows in the first table multiplied by the number of rows in the second table if nowhere clause is used along with Cross join. This is also called a Cartesian product.

If where clause is used with Cross join, it functions like Inner join

Syntax: Select * from table1 cross join table2.

Eg: Select * from doc 1 Cross join doc2.

c) Self Join

A self-join is a regular join that select table is joined with itself.

Syntax: Select(column_names) from table1 T1, table T2 where condition;

d) Inner Join

The inner join keyword selects records that have matching values in both tables.

Syntax: Select (column_name) from <Table1> inner join <Table2> on table1.colum_name = table2.column_name;

Eg: Select * from doc1 inner join doc2 on doc1.gender = doc2.gender.

e) Outer Join

An outer join makes one of the tables dominant such a table is called the outer table and the other table is called a subordinate table.

In an outer join, the resultant table contains the combination of rows from the dominant table that satisfies the join condition and also rows that do not have matching rows in the subordinate table.

It is of three types:

1) Left Join
2) Right Join
3) Full Join

1) Left Join

The left join keyword returns all records from the left table (table1) and the matched records from the right table (table2).

Syntax: Select (column_names) from (table1) left join (table2) on table1.column_name = table2.column_name.

Eg: Select * from doc1 left join doc2 on doc1.id = doc2.id

2) Right Join

The right join keyword returns all records from the right table (table2) and the matched records from the left table (table1).

Syntax: Select (column_names) from (table1) right outer join (table2) on table1.column_name = table2.column_name;

Eg: Select * from doc1 right join doc2 on doc1.id = doc2.id.

3) **Full Join**

The full outer join keyword returns all records when there is a match in left (table1) or right (table2) table records.

Syntax: Select (column_names) from (table1) full outer join (table2) on table1.colum_name = table2.column_name where condition;

Eg: Select * from doc1 full join doc2 on doc1.id = doc2.id.

Conclusion: Query to write joins are implemented and studied successfully.

EXPERIMENT NO 4: JOIN COMMANDS

NAME: ANANYA **DATE:** 17/2/2020 **ROLL NO:** 4

QUERY: Show existing data in table ‘complaints’ & ‘reviews’

SQL Worksheet

```
1 insert into complaints values('c20201','bad service','202001');
2 insert into complaints values('c20202','bad staff','202002');
3 insert into complaints values('c20203','slow service','202003');
4
5 select * from complaints;
```

COMPLAINT_ID	COMPLAINT_DESC	ID
c20201	bad service	202001
c20202	bad staff	202002
c20203	slow service	202003

Download CSV
3 rows selected.

SQL Worksheet

```
1 insert into reviews values('202001','fast service','1');
2 insert into reviews values('202002','good service','2');
3 insert into reviews values('202004','good staff','3')
4
5 select * from reviews;
```

ID	REVIEW_DESC	REVIEW_LEVEL
202001	fast service	1
202002	good service	2
202004	good staff	3

Download CSV
3 rows selected.

QUERY: Natural join complaints & reviews table

SQL Worksheet

Clear Find Actions Save Run

```
1 select * from complaints natural join reviews;
```

ID	COMPLAINT_ID	COMPLAINT_DESC	REVIEW_DESC	REVIEW_LEVEL
202001	c20201	bad service	fast service	1
202002	c20202	bad staff	good service	2

Download CSV
2 rows selected.

QUERY: Cross join complaints & reviews table

SQL Worksheet

Clear Find Actions Save Run

```
1  select *  from complaints cross join reviews;
```

COMPLAINT_ID	COMPLAINT_DESC	ID	ID	REVIEW_DESC	REVIEW_LEVEL
c20201	bad service	202001	202001	fast service	1
c20201	bad service	202001	202002	good service	2
c20201	bad service	202001	202004	good staff	3
c20202	bad staff	202002	202001	fast service	1
c20202	bad staff	202002	202002	good service	2
c20202	bad staff	202002	202004	good staff	3
c20203	slow service	202003	202001	fast service	1
c20203	slow service	202003	202002	good service	2
c20203	slow service	202003	202004	good staff	3

Download CSV
9 rows selected.

QUERY: Implement self join on table ‘complaints’

SQL Worksheet

Clear Find Actions Save Run

```
1  select c1.*, c2.* from complaints c1, complaints c2 where c1.id = c2.id;
```

COMPLAINT_ID	COMPLAINT_DESC	ID	COMPLAINT_ID	COMPLAINT_DESC	ID
c20201	bad service	202001	c20201	bad service	202001
c20202	bad staff	202002	c20202	bad staff	202002
c20203	slow service	202003	c20203	slow service	202003

Download CSV
3 rows selected.

QUERY: Inner join complaints & reviews table

SQL Worksheet

Clear | Find | Actions | Save | Run

```
1  select * from complaints inner join reviews on complaints.id=reviews.id;
```

COMPLAINT_ID	COMPLAINT_DESC	ID	ID	REVIEW_DESC	REVIEW_LEVEL
c20201	bad service	202001	202001	fast service	1
c20202	bad staff	202002	202002	good service	2

Download CSV

2 rows selected.

QUERY: Left join complaints & reviews table

SQL Worksheet

Clear | Find | Actions | Save | Run

```
1  select * from complaints left join reviews on complaints.id=reviews.id;
```

COMPLAINT_ID	COMPLAINT_DESC	ID	ID	REVIEW_DESC	REVIEW_LEVEL
c20201	bad service	202001	202001	fast service	1
c20202	bad staff	202002	202002	good service	2
c20203	slow service	202003	–	–	–

Download CSV

3 rows selected.

QUERY: Right join complaints & reviews table

SQL Worksheet

Clear Find Actions Save Run

```
1  select * from complaints left join reviews on complaints.id=reviews.id;
```

COMPLAINT_ID	COMPLAINT_DESC	ID	ID	REVIEW_DESC	REVIEW_LEVEL
c20201	bad service	202001	202001	fast service	1
c20202	bad staff	202002	202002	good service	2
c20203	slow service	202003	-	-	-

Download CSV

3 rows selected.

QUERY: Full join complaints & reviews table

SQL Worksheet

Clear Find Actions Save Run

```
1  select * from complaints full join reviews on complaints.id=reviews.id;
```

COMPLAINT_ID	COMPLAINT_DESC	ID	ID	REVIEW_DESC	REVIEW_LEVEL
c20201	bad service	202001	202001	fast service	1
c20202	bad staff	202002	202002	good service	2
-	-	-	202004	good staff	3
c20203	slow service	202003	-	-	-

Download CSV

4 rows selected.

EXPERIMENT NO : 5

AGGREGATE FUNCTION

Aim: Write the query for implementing the following aggregate functions: max(), min(), avg(), sum(), count() and clauses like group by and order by.

Theory: Aggregate functions: It takes a collection of values and returns a single value result.

1. COUNT() : It returns the number of values in specified field. It can work on both numeric and nom-numeric data-types.

Syntax: SELECT COUNT(*) OR COUNT (COLUMN_NAME)

FROM <TABLE_NAME>

EG: SELECT GENDER ,COUNT(*) FROM PATIENT

SELECT COUNT(DOC ID, DOC NAME) FROM DOCTOR

SELECT COUNT(GENDER) FROM DOCTOR WHERE GENDER= F

2. MIN() : It returns the smallest value specified in the table field. It is used to find out the minimum value of a certain column.

Syntax: SELECT MIN (COLUMN_NAME) FROM <TABLE_NAME>

EG: SELECT MIN(SALARY) FROM ACTORS

3.MAX() : It returns the maximum value specified in the table field. It is used to find out the maximum value of a certain column.

Syntax: SELECT MAX (COLUMN_NAME) FROM <TABLE_NAME>

EG: SELECT MAX(SALARY) FROM ACTORS

4. SUM() : It is used to calculate sum of the selected column. Used on numeric fields only.

Syntax: SELECT SUM (COLUMN_NAME) FROM <TABLE_NAME>

EG: SELECT SUM(SALARY) FROM ACTORS

5.AVG() : It is used to calculate sum of the selected column. Used on numeric fields only.

Syntax: SELECT AVG (COLUMN_NAME) FROM <TABLE_NAME>

EG: SELECT AVG(COST) FROM PRODUCTION

WHERE CLAUSE:

It is used to filter records. To extract only those records which fulfil certain criteria

SYNTAX: SELECT (COL1, COL2,…..) FROM <TABLE_NAME>

WHERE CONDITION;

EG: SELECT * FROM NAMES WHERE YEAR = ’2000’;

HAVING CLAUSE:

It is used because where clause cannot be used with aggregate functions

SYNTAX: SELECT (COL1, COL2,…..) FROM <TABLE_NAME>

HAVING CONDITION;

EG: SELECT (MOVIE_NAME, ACTORS) FROM MOVIES

GROUP BY MOVIE_NAME, ACTORS, YEAR HAVING YEAR> 2001;

Group by clause

It groups rows that have same values into summary rows like “find the number of customer in each country”. It is often used with aggregate functions to group the result cell by one or more columns.

SYNTAX: - SELECT (COLUMN_NAME)

FROM (TABLE_NAME)

WHERE CONDITION GROUP BY (COLUMN_NAME);

EXAMPLE:-SELECT COUNT (CUSTOMER_ID), COUNTRY

FROM CUSTOMERS

GROUP BY COUNTRY

Order by clause

It is used to sort the results in ascending or descending. It sorts in ascending order by default.

SYNTAX: -SELECT (COLUMN_NAME)

FROM (TABLE_NAME)

ORDER BY (COLUMN_NAME) ASC/DESC;

EXAMPLE: -SELECT * FROM CUSTOMERS ORDER BY COUNTRY;

SELECT * FROM CUSTOMERS ORDER BY CUSTOMER_NAME DESC;

Conclusion: - Functions like min(), max(), avg(), sum(), count(), and clauses like where, having, group by, Order by are implemented.

EXPERIMENT NO 5: AGGREGATE FUNCTIONS

NAME: ANANYA **DATE:** 10/2/2020 **ROLL NO:** 4

QUERY: Show existing data in table ‘employee’ & ‘material’

SQL Worksheet

```
1 select * from employee;
```

EMP_ID	EMP_NAME	EMP_CONTACT	EMP_DESIGNATION	EMP_ADDRESS	EMP_SALARY	EMP_GENDER	EMP_EMAIL
e20203	ananya	9768132727	founder	bandra	5000	m	ananya@gmail.com
e20204	junaid	8828357885	manager	dadar	400	m	junaid@gmail.com
e20205	pranav	9819549376	intern	grant road	30	m	pranav@gmail.com
e20202	hasan	9833498962	cfo	malad	500	m	charolya@gmail.com
e20201	nivaan	9930601132	ceo	kharghar	1000	m	nivaang@gmail.com

Download CSV
5 rows selected.

SQL Worksheet

```
1 select * from material;
```

MAT_QUALITY	MAT_LENGTH	MAT_NAME
bad	32	cotton
average	55	cashmere
good	22	silk
good	25	silk

Download CSV
4 rows selected.

QUERY: Find the employee with the minimum salary in the ‘employee’ table

SQL Worksheet

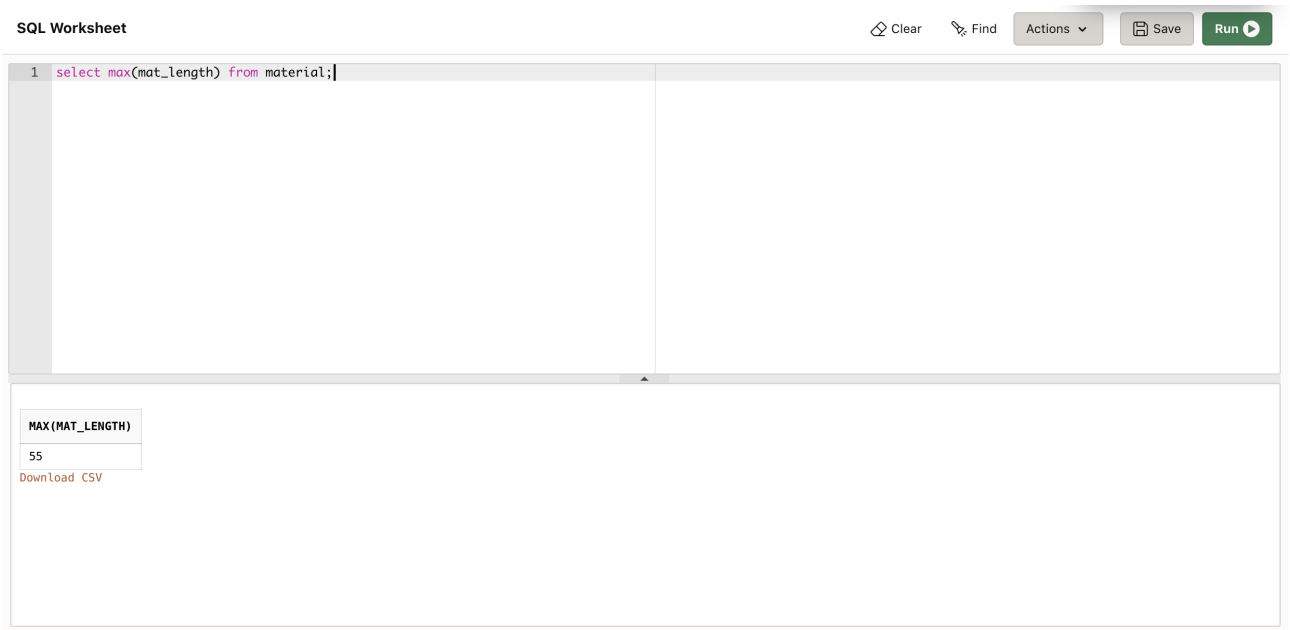
Clear Find Actions Save Run

```
1 select min(emp_salary) from employee;
```

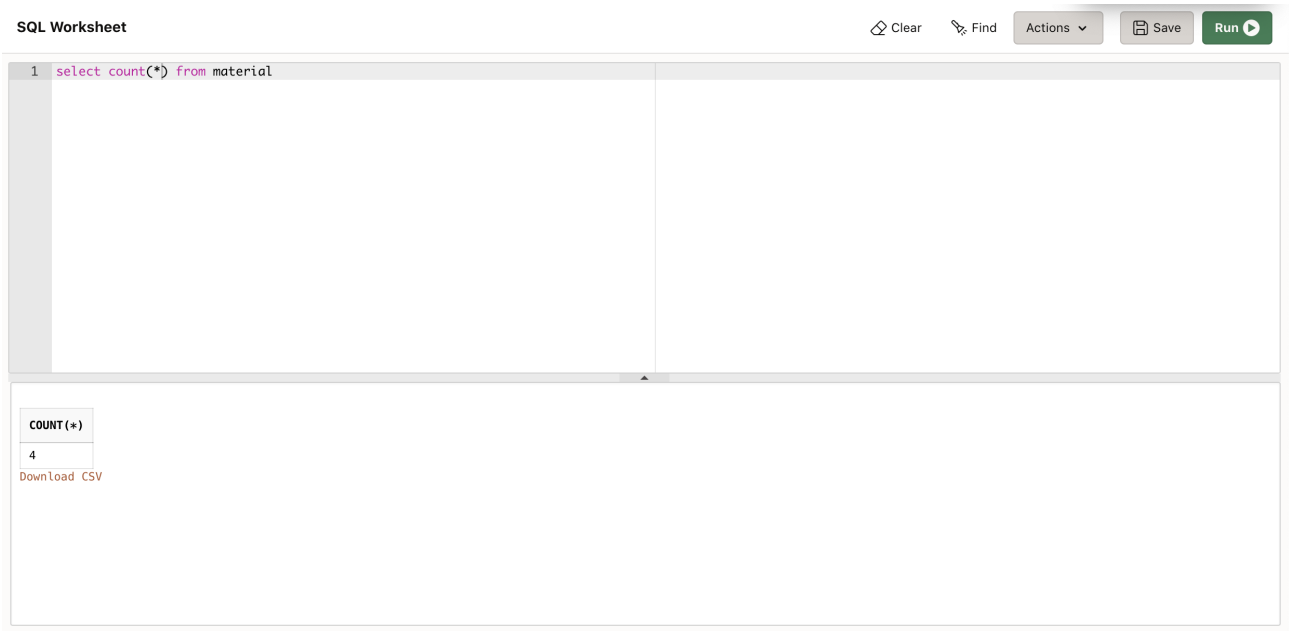
MIN(EMP_SALARY)
30

Download CSV

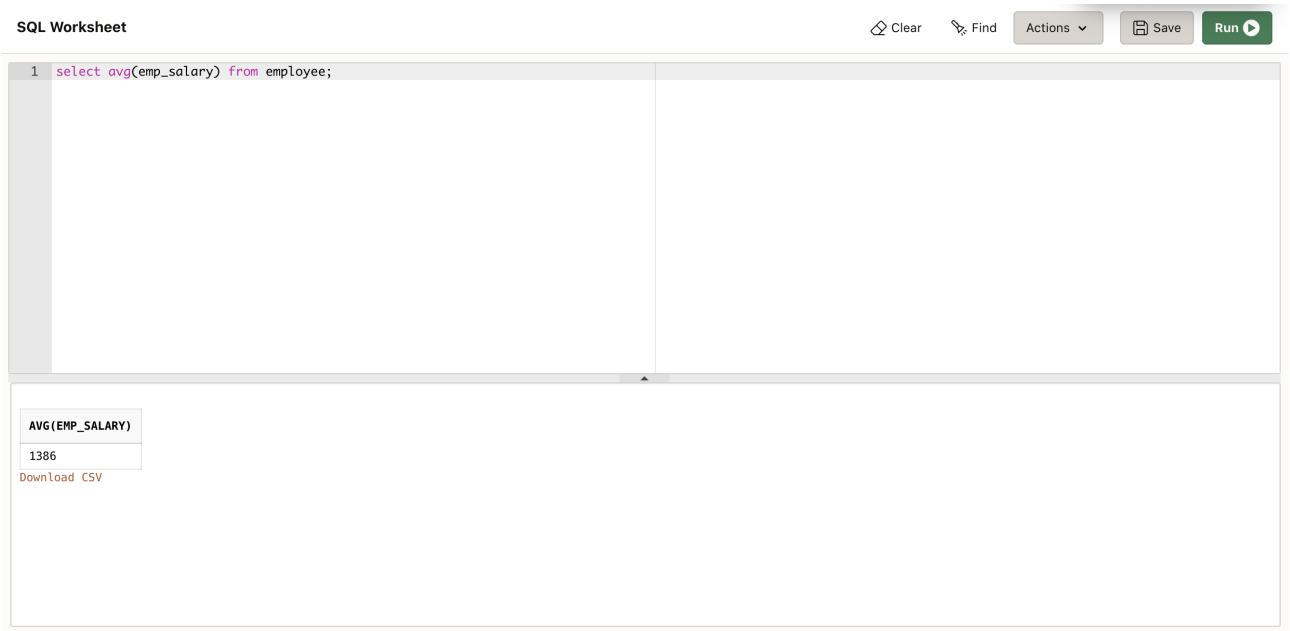
QUERY: Find the material with the maximum length from ‘material’ table



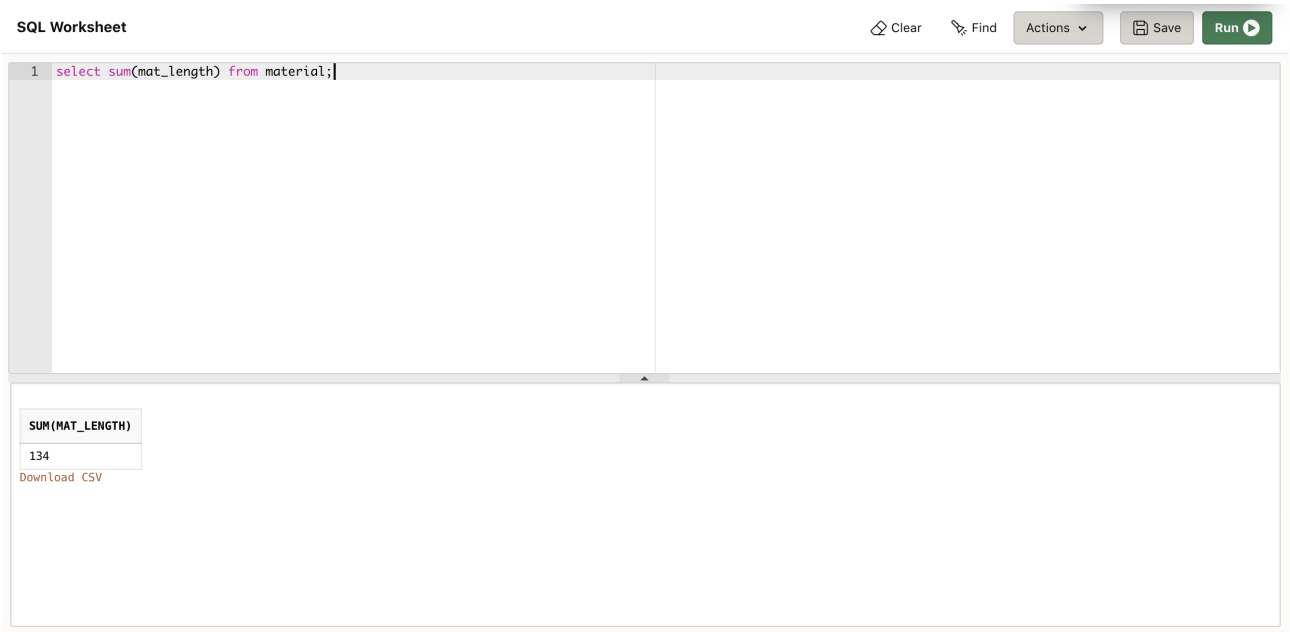
QUERY: Find the count of the rows in the table ‘material’



QUERY: Find the average salary of employees from ‘employee’ table



QUERY: Find the sum of material lengths from the ‘material’ table



QUERY: Use all the aggregate functions on table employee

SQL Worksheet

Clear Find Actions Save Run

```
1  select min(emp_salary), max(emp_salary), avg(emp_salary), sum(emp_salary), count(emp_salary) from employee;
```

MIN(EMP_SALARY)	MAX(EMP_SALARY)	AVG(EMP_SALARY)	SUM(EMP_SALARY)	COUNT(EMP_SALARY)
30	5000	1386	6930	5

Download CSV

EXPERIMENT NO : 6

VIEWS

Aim: Write the queries for creating, updating and deleting views in SQL.

Theory: Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain condition.

1. **CREATING VIEWS:** We can create View using CREATE VIEW statement. A View can be created from a single table or multiple tables.

   Syntax:
   ```
   CREATE VIEW view_name AS
   SELECT column1, column2.....
   FROM table_name
   WHERE condition;
   ```

   Examples:

   - **Creating View from a single table:**
     In this example we will create a View named DetailsView from the table StudentDetails.

     Query:
     ```
     CREATE VIEW DetailsView AS
     SELECT NAME, ADDRESS
     FROM StudentDetails
     WHERE S_ID < 5;
     ```

   - **Creating View from multiple tables:**
     In this example we will create a View named MarksView from two tables StudentDetails and StudentMarks. To create a View from multiple tables we can simply include multiple tables in the SELECT statement.

     Query:
     ```
     CREATE VIEW MarksView AS
     SELECT StudentDetails.NAME, StudentDetails.ADDRESS, StudentMarks.MARKS
     FROM StudentDetails, StudentMarks
     WHERE StudentDetails.NAME = StudentMarks.NAME;
     ```

   Horizontal view: It allows the user to access the tuples which satisfy the WHERE clause.

   Example:

   ```
   CREATE VIEW PASSED AS
   SELECT *
   ```

```
FROM STUDENTS
WHERE RESULT='P'
```

Vertical view: It restricts user's access to certain columns of the table.

Example:

```
CREATE VIEW DETAILS AS
SELECT NAME, ADDRESS
FROM StudentDetails
```

Row/Column subset view: It is a combination of horizontal & vertical views.

Example:

```
CREATE VIEW DetailsView AS
SELECT NAME, ADDRESS
FROM StudentDetails
WHERE S_ID < 5;
```

2. **UPDATING VIEWS:** There are certain conditions needed to be satisfied to update a view. If any one of these conditions is not met, then we will not be allowed to update the view.

- The SELECT statement which is used to create the view should not include GROUP BY clause or ORDER BY clause.
- The SELECT statement should not have the DISTINCT keyword.
- The View should have all NOT NULL values.
- The view should not be created using nested queries or complex queries.
- The view should be created from a single table. If the view is created using multiple tables then we will not be allowed to update the view.

We can use the CREATE OR REPLACE VIEW statement to add or remove fields from a view.

Syntax:

```
CREATE OR REPLACE VIEW view_name AS

SELECT column1,coulmn2,..

FROM table_name

WHERE condition;
```

Example: If we want to update the view MarksView and add the field AGE to this View from StudentMarks Table, we can do this as:

```
CREATE OR REPLACE VIEW MarksView AS

SELECT StudentDetails.NAME, StudentDetails.ADDRESS, StudentMarks.MARKS,
StudentMarks.AGE
```

```
FROM StudentDetails, StudentMarks
WHERE StudentDetails.NAME = StudentMarks.NAME;
```

3. **DELETING VIEWS:** We can delete or drop a View using the DROP statement.

   Syntax:

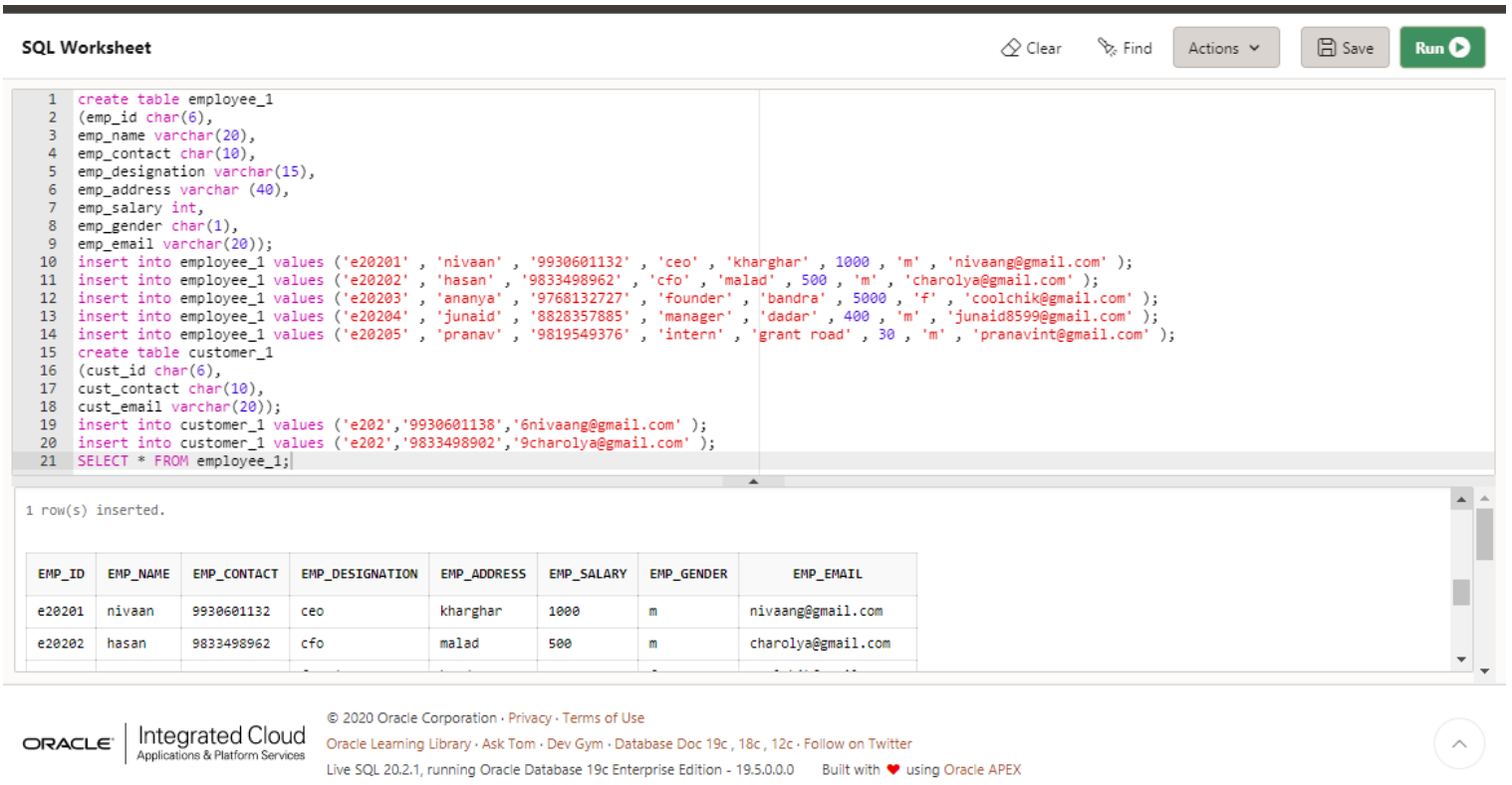
   DROP VIEW view_name;

   Example: If we want to delete the View MarksView, we can do this as:

   DROP VIEW MarksView;

Conclusion: - Operations like creation, updation, and deletion of views were implemented.

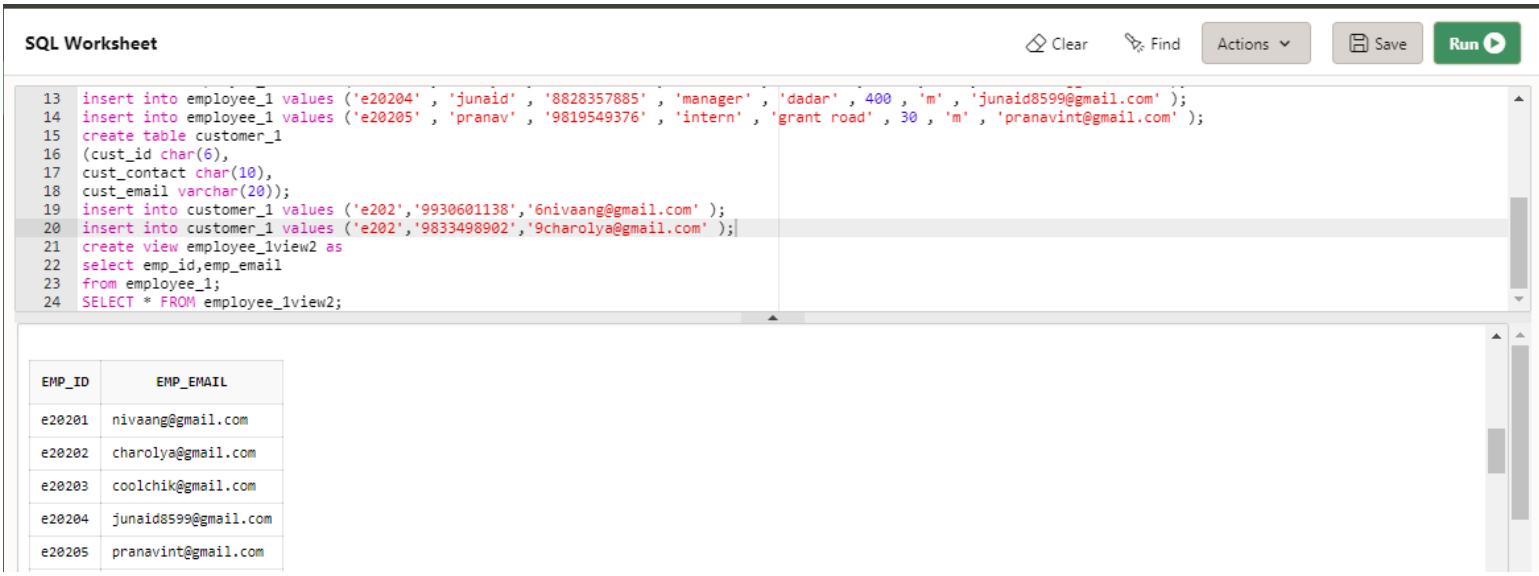
This is showing the data rows and columns of TABLE employee_1



This is showing the data rows and columns of both TABLE employee_1 and customer_1

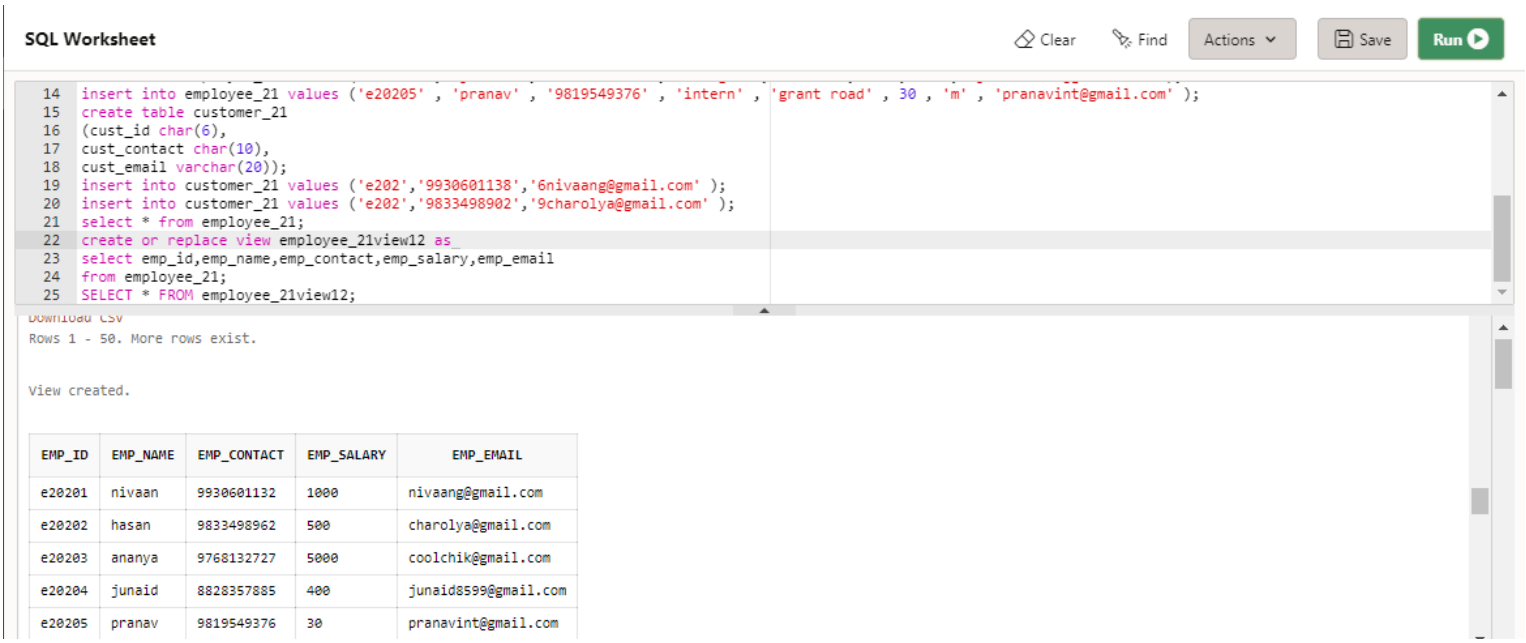
This is showing the data rows and columns of view employee_1view: columns - id, name and email only.

This is showing the data rows and columns of view employee_1view2: columns – id and email only.

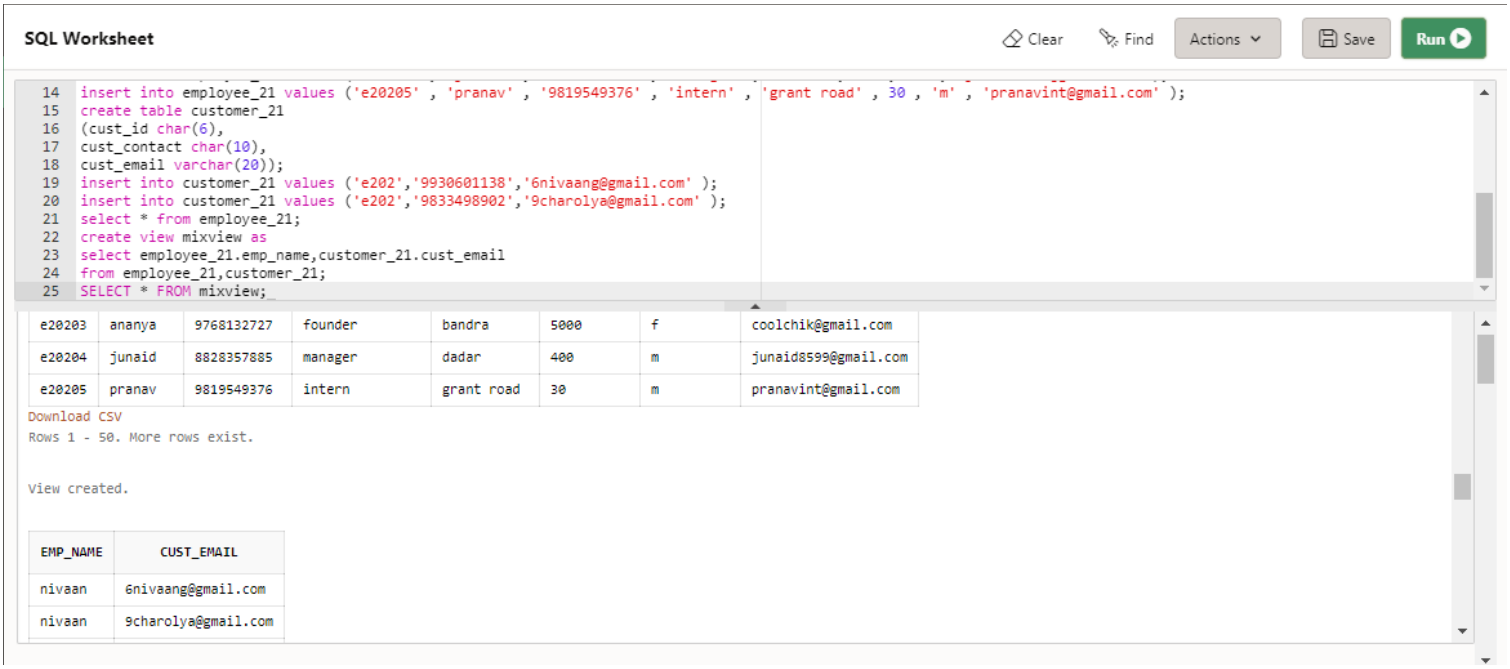


This is showing the data rows and columns of VIEW employee_1view12 : columns - id, name, salary and email only whose salary = 30.

This is showing the data rows and columns of the replaced view employee_21view12 : columns - id, name, contact, salary and email only.



View mixview created.



VIEW mixview DROPPED.

Experiment 7

Implement TCL commands

Aim: To implement Transaction Control Language (TCL) commands

Theory: Transaction Control Language is generally used to save changes done in data into table of database or undo all unnecessary changes. These are used to manage transactions in the database and manage changes made by DML statements. It also allows statements to be grouped together into logical transactions.

TCL commands are:

a. a. Commit- It ends your current transaction and makes all the changes performed permanent. It also erases all save points in the transaction and releases the transaction locks
Syntax: commit;

b. b. Rollback- It is used to restore the database to the last committed state. Additionally, it is also used with save point command for jumping to a save point in a transaction. If save point is not mentioned, the entire transaction is rolled back.
Syntax: rollback to <savepoint_name>

c. c. Save Point- Its main use is to save a transaction temporarily. This way users can rollback to the point whenever it is needed. An active save point since the last commit or rollback
Syntax: savepoint <savepoint_name>

Queries

With below table:

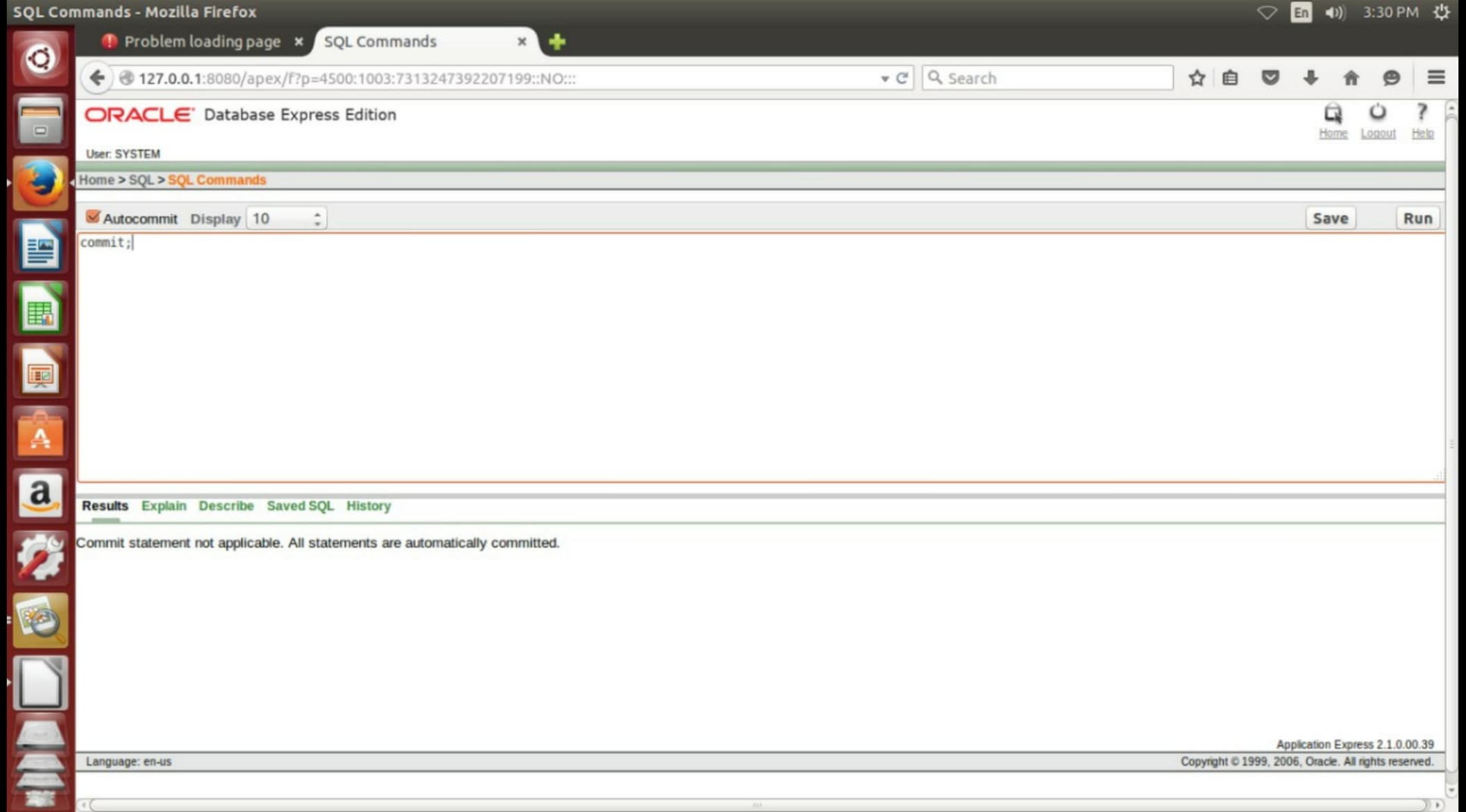
ID	Name
98	Anita
99	Maria

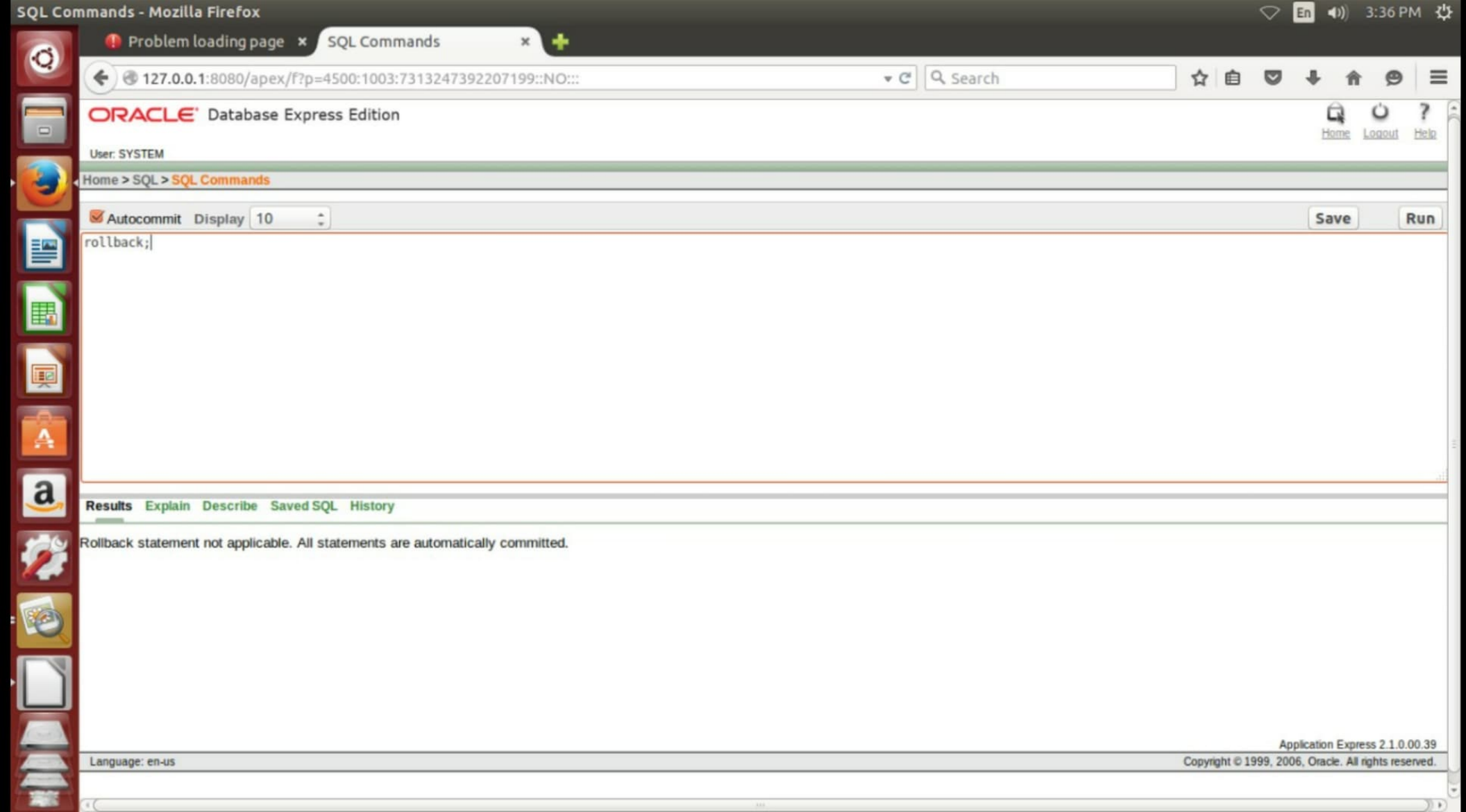
- \ Insert into class values (100, 'Rahul');
- \ Commit;
- \ Update class set name = 'Tyler' where id=100
- \ Savepoint A;
- \ Insert into class values(101, 'zack');
- \ Savepoint b;
- \ Rollback to A;
- \ Select * from class

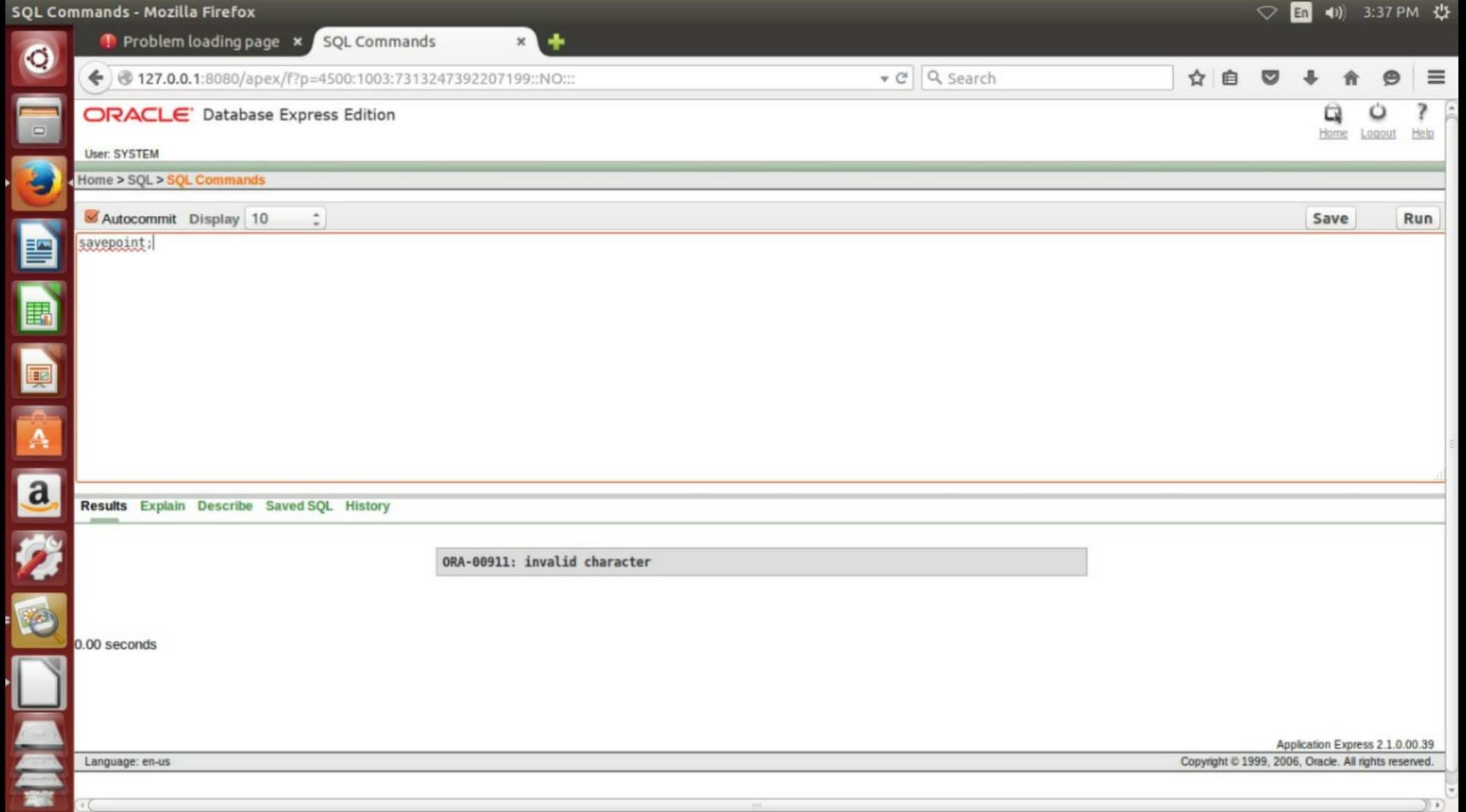
Output:

ID	Name
98	Anita
99	Maria
100	Tyler

Conclusion: TCL commands are implemented successfully


SQL Commands - Mozilla Firefox
En
3:30 PM
Problem loading page
SQL Commands
127.0.0.1:8080/apex/f?p=4500:1003:7313247392207199::NO:::
Search
ORACLE Database Express Edition
Home
Logout
Help
User: SYSTEM
Home > SQL > SQL Commands
Autocommit
Display 10
Save
Run
commit;
Results
Explain
Describe
Saved SQL
History
Commit statement not applicable. All statements are automatically committed.
Application Express 2.1.0.00.39
Language: en-us
Copyright © 1999, 2006, Oracle. All rights reserved.


SQL Commands - Mozilla Firefox
En
3:36 PM
Problem loading page
SQL Commands
127.0.0.1:8080/apex/f?p=4500:1003:7313247392207199::NO:::
Search
ORACLE Database Express Edition
Home
Logout
Help
User: SYSTEM
Home > SQL > SQL Commands
Autocommit
Display 10
Save
Run
rollback;
Results
Explain
Describe
Saved SQL
History
Rollback statement not applicable. All statements are automatically committed.
Application Express 2.1.0.00.39
Language: en-us
Copyright © 1999, 2006, Oracle. All rights reserved.


SQL Commands - Mozilla Firefox
En
3:37 PM
Problem loading page
SQL Commands
127.0.0.1:8080/apex/f?p=4500:1003:7313247392207199::NO:::
Search
ORACLE Database Express Edition
Home
Logout
Help
User: SYSTEM
Home > SQL > SQL Commands
Autocommit
Display 10
Save
Run
savepoint;
Results
Explain
Describe
Saved SQL
History
ORA-00911: invalid character
0.00 seconds
Application Express 2.1.0.00.39
Language: en-us
Copyright © 1999, 2006, Oracle. All rights reserved.

EXPERIMENT NO-8

Aim- To study Procedural Language extensions to the Structured Query Language and implement commands in pl sql.

Theory - Procedural Language extensions to the Structured Query Language adds many procedural constructs to SQL language to overcome some limitations of SQL. Besides, PL/SQL provides a more comprehensive programming language solution for building mission-critical applications on Oracle Databases.

In PL/SQL, the code is not executed in single line format, but it is always executed by grouping the code into a single element called Blocks.Blocks contain both PL/SQL as well as SQL instruction. All these instruction will be executed as a whole rather than executing a single instruction at a time. PL/SQL blocks can include variables, SQL statements, loops, constants, conditional statements and exception handling. Blocks can also build a function or a procedure or a package.

PL/SQL blocks are two types:

1) **Anonymous blocks:** In PL/SQL, That's blocks which is not have header are known as anonymous blocks. These blocks do not form the body of a function or triggers or procedure.

   Characteristics of Anonymous blocks-
   - These blocks don't have any reference name specified for them.
   - These blocks start with the keyword 'DECLARE' or 'BEGIN'.
   - Since these blocks do not have any reference name, these cannot be stored for later purpose. They shall be created and executed in the same session.
   - They can call the other named blocks, but call to anonymous block is not possible as it is not having any reference.
   - It can have nested block in it which can be named or anonymous. It can also be nested in any blocks.
   - These blocks can have all three sections of the block, in which execution section is mandatory, the other two sections are optional.

2) **Named blocks:** That's PL/SQL blocks which having header or labels are known as Named blocks. These blocks can either be subprograms like functions, procedures, packages or Triggers.

   Characteristics of Named blocks.
   - These blocks can be called from other blocks.

- The block structure is same as an anonymous block, except it will never start with the keyword 'DECLARE'. Instead, it will start with the keyword 'CREATE' which instruct the compiler to create it as a database object.
- These blocks can be nested within other blocks. It can also contain nested blocks.

Block structure

PL/SQL blocks have a pre-defined structure in which the code is to be grouped. Below are different sections of PL/SQL blocks.

1. Declaration section-This is the first section of the PL/SQL blocks. This section is an optional part. This is the section in which the declaration of variables, cursors, exceptions, subprograms, pragma instructions and collections that are needed in the block will be declared.
2. Execution section- Execution part is the main and mandatory part which actually executes the code that is written inside it. Since the PL/SQL expects the executable statements from this block this cannot be an empty block, i.e., it should have at least one valid executable code line in it.
3. Exception-Handling section- The exception is unavoidable in the program which occurs at run-time and to handle this Oracle has provided an Exception-handling section in blocks. This section can also contain PL/SQL statements. This is an optional section of the PL/SQL blocks.

Syntax -

```
DECLARE --optional

   <declarations>


BEGIN   --mandatory

   <executable statements. At least one executable statement is mandatory>


EXCEPTION --optional

   <exception handles>
```

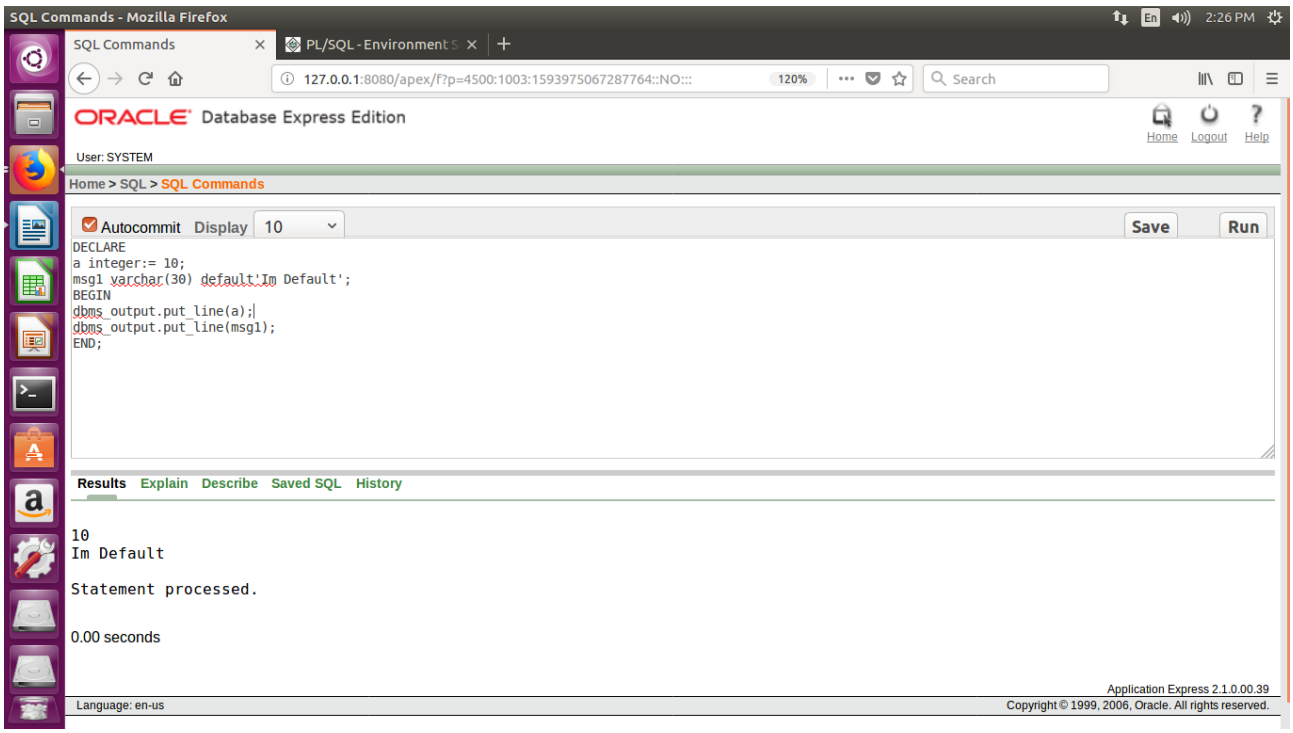
```
END;   --mandatory
/
```

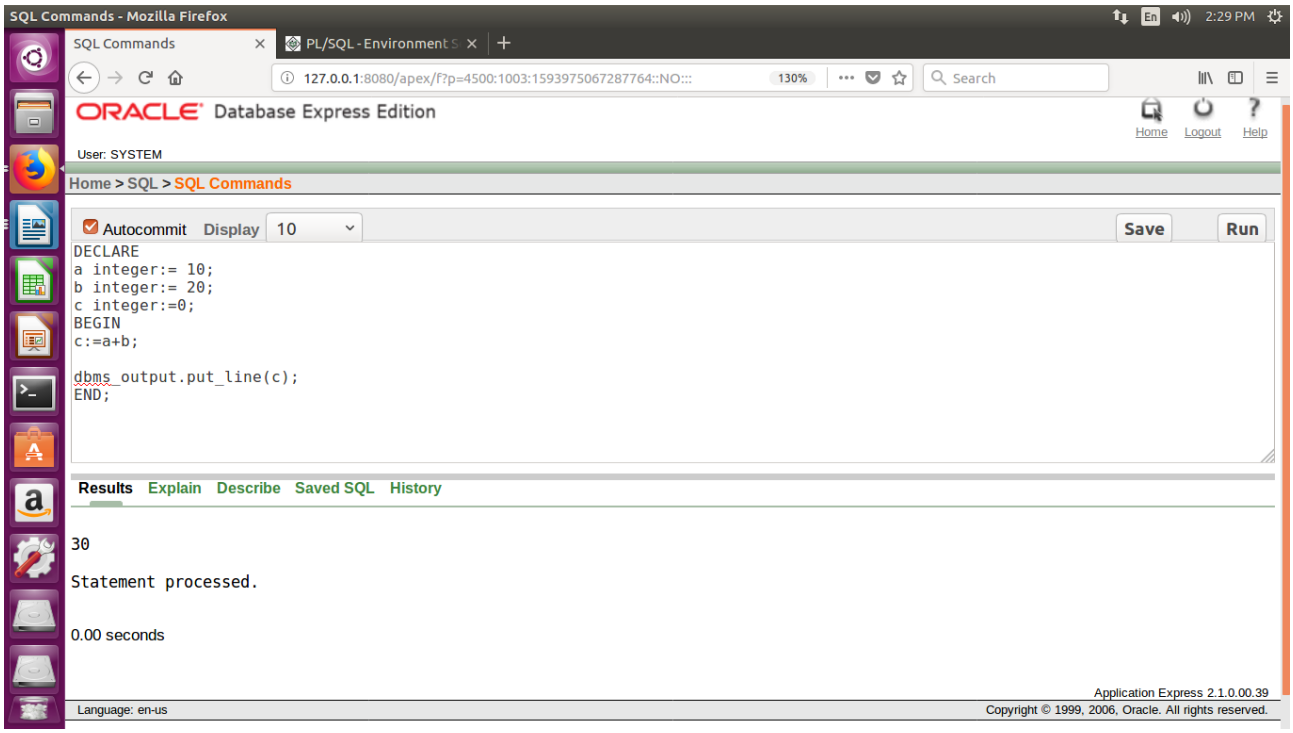
A block should always be followed by '/' which sends the information to the compiler about the end of the block.

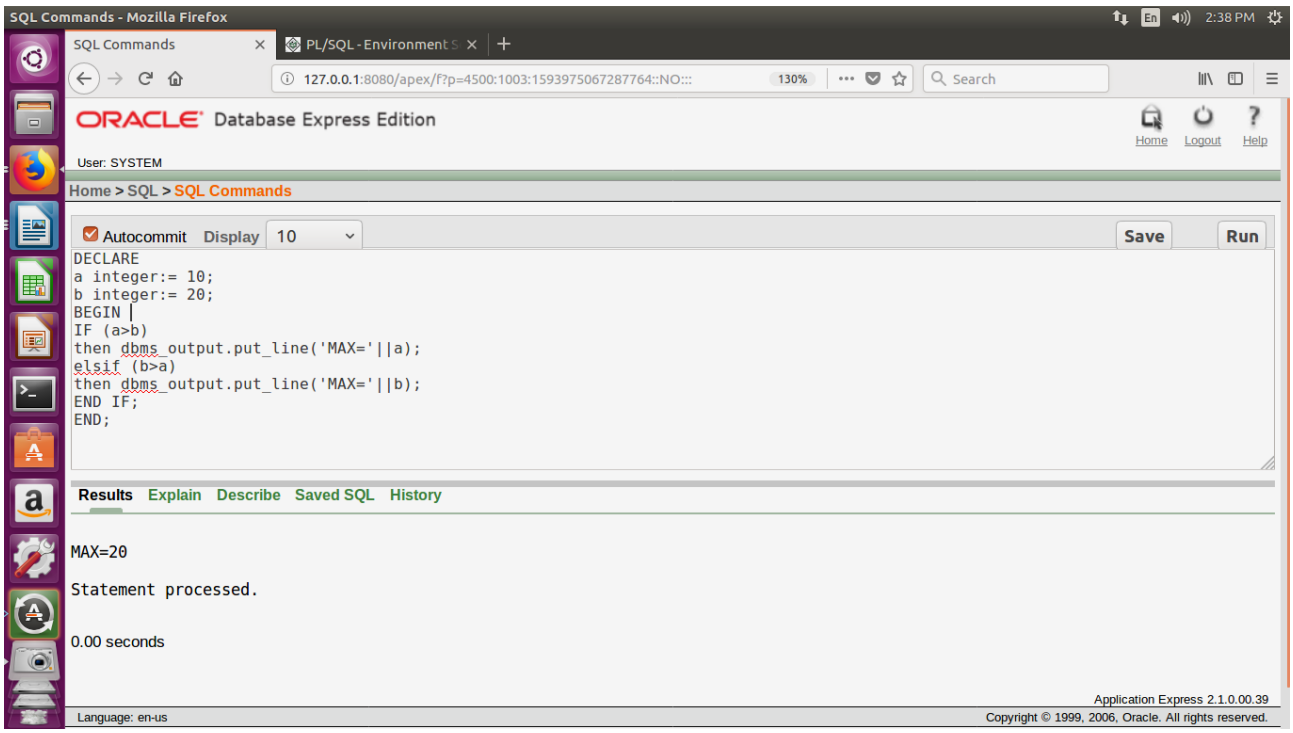
Example -To print even numbers between 0 to 100-

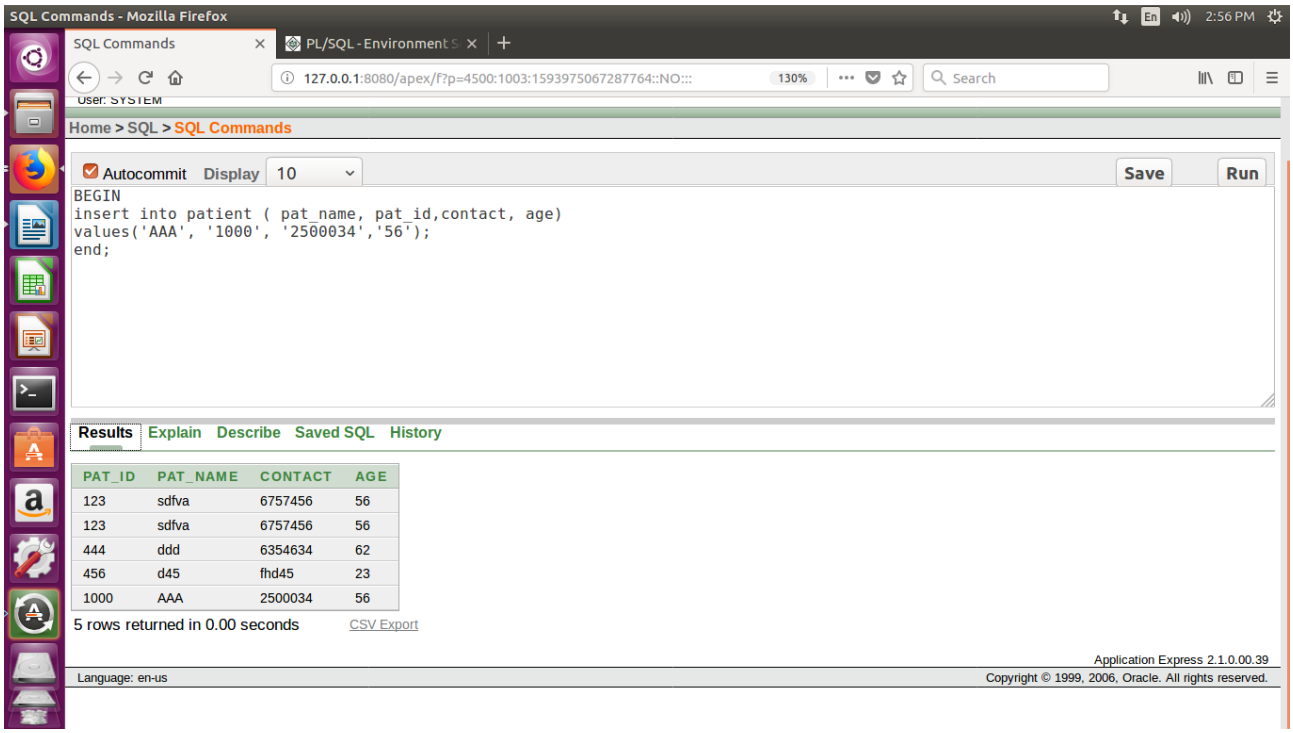
```
Declare
NUM1 number:=0;
begin
loop
NUM1 := NUM1+2;
dbms_output.put_line (NUM1||',');
exit when NUM1=100;
end loop;
End;
```

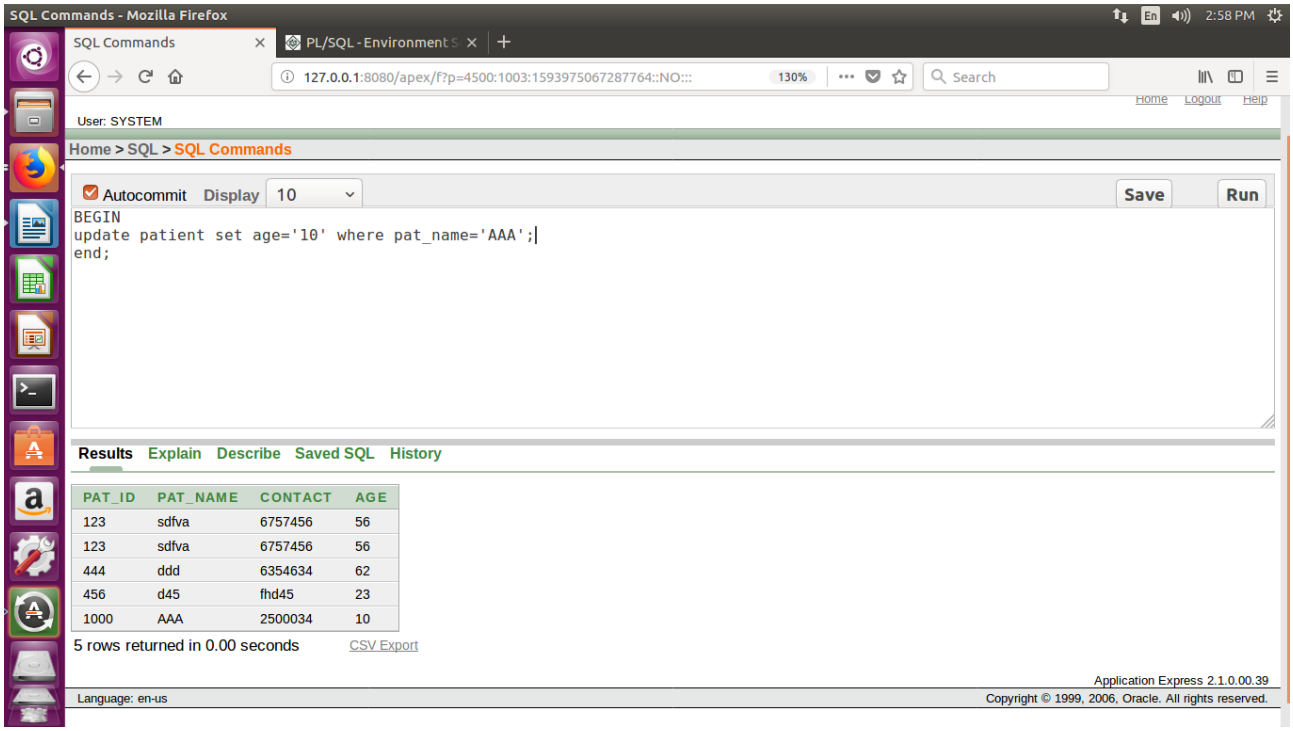
Conclusion- Different types of PL/SQL blocks are learned along with syntax.

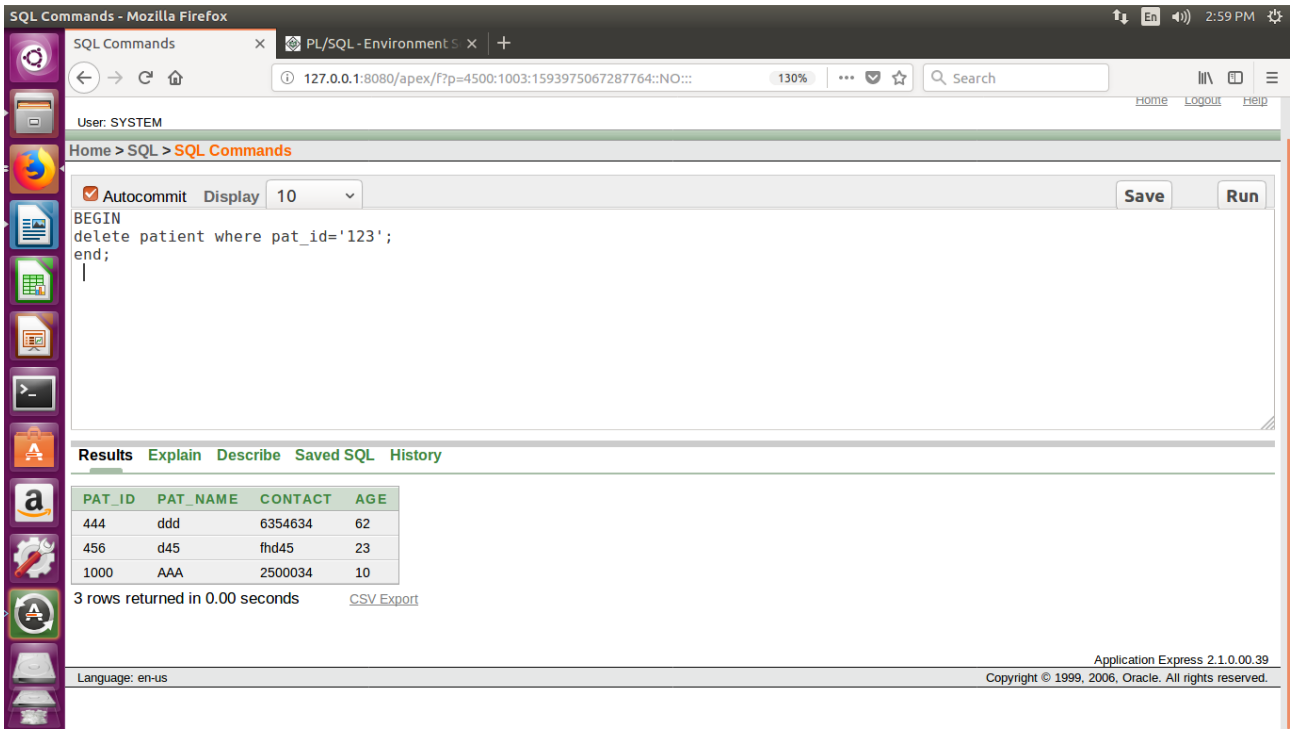

SQL Commands - Mozilla Firefox
2:26 PM
SQL Commands
PL/SQL - Environment
127.0.0.1:8080/apex/f?p=4500:1003:1593975067287764::NO:::
120%
Search
ORACLE Database Express Edition
Home
Logout
Help
User: SYSTEM
Home > SQL > SQL Commands
Autocommit
Display
10
Save
Run
DECLARE
a integer:= 10;
msg1 varchar(30) default'Im Default';
BEGIN
dbms_output.put_line(a);
dbms_output.put_line(msg1);
END;
Results
Explain
Describe
Saved SQL
History
10
Im Default
Statement processed.
0.00 seconds
Application Express 2.1.0.00.39
Language: en-us
Copyright © 1999, 2006, Oracle. All rights reserved.

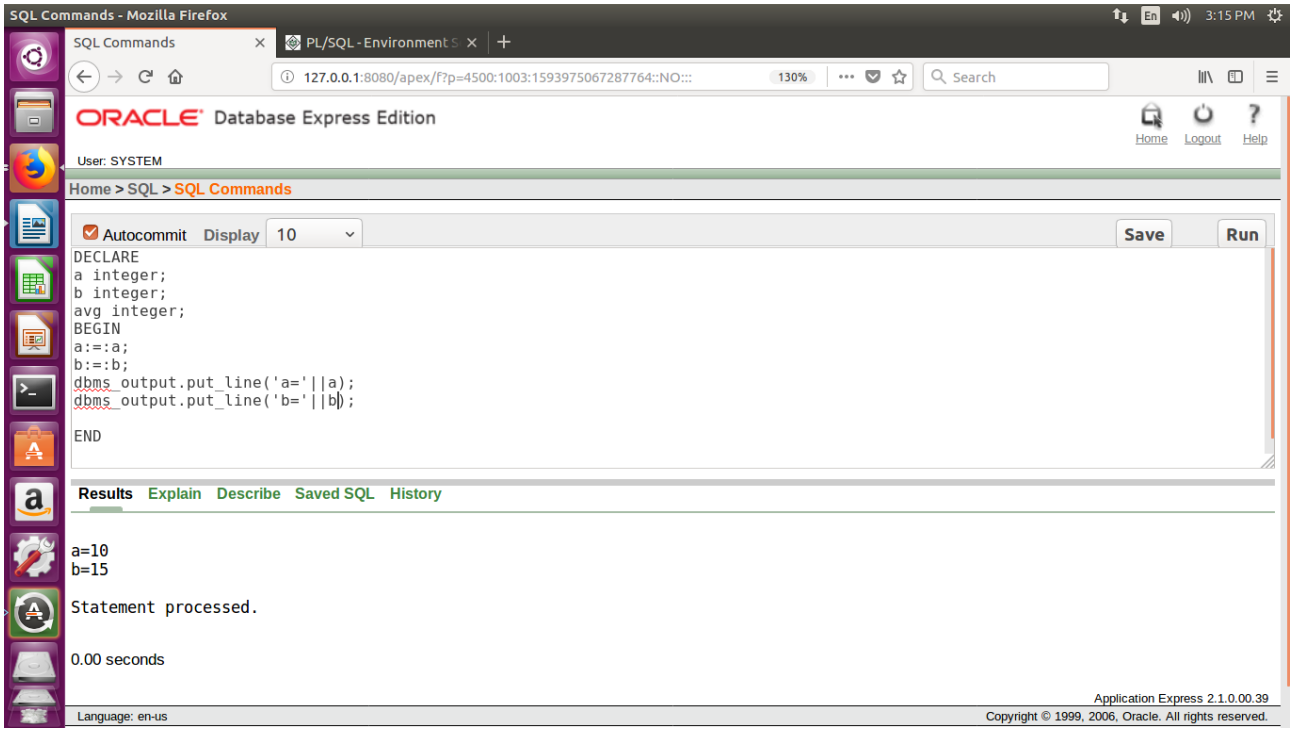

SQL Commands - Mozilla Firefox
2:29 PM
SQL Commands
PL/SQL - Environment
127.0.0.1:8080/apex/f?p=4500:1003:1593975067287764::NO:::
130%
Search
ORACLE Database Express Edition
Home
Logout
Help
User: SYSTEM
Home > SQL > SQL Commands
Autocommit
Display
10
Save
Run
DECLARE
a integer:= 10;
b integer:= 20;
c integer:=0;
BEGIN
c:=a+b;
dbms_output.put_line(c);
END;
Results
Explain
Describe
Saved SQL
History
30
Statement processed.
0.00 seconds
Application Express 2.1.0.00.39
Language: en-us
Copyright © 1999, 2006, Oracle. All rights reserved.

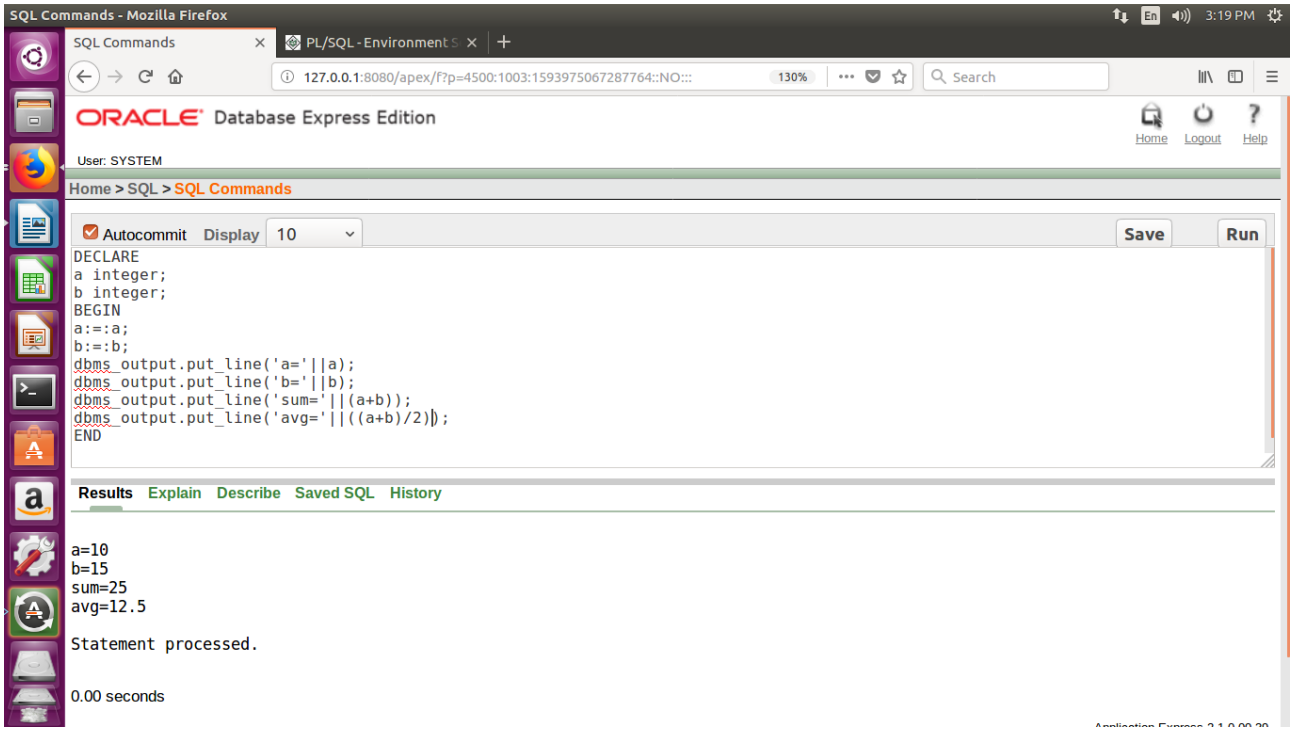

SQL Commands - Mozilla Firefox
2:38 PM
SQL Commands
PL/SQL - Environment
127.0.0.1:8080/apex/f?p=4500:1003:1593975067287764::NO:::
130%
Search
ORACLE Database Express Edition
Home
Logout
Help
User: SYSTEM
Home > SQL > SQL Commands
Autocommit
Display
10
Save
Run
DECLARE
a integer:= 10;
b integer:= 20;
BEGIN
IF (a>b)
then dbms_output.put_line('MAX='||a);
elsif (b>a)
then dbms_output.put_line('MAX='||b);
END IF;
END;
Results
Explain
Describe
Saved SQL
History
MAX=20
Statement processed.
0.00 seconds
Application Express 2.1.0.00.39
Language: en-us
Copyright © 1999, 2006, Oracle. All rights reserved.

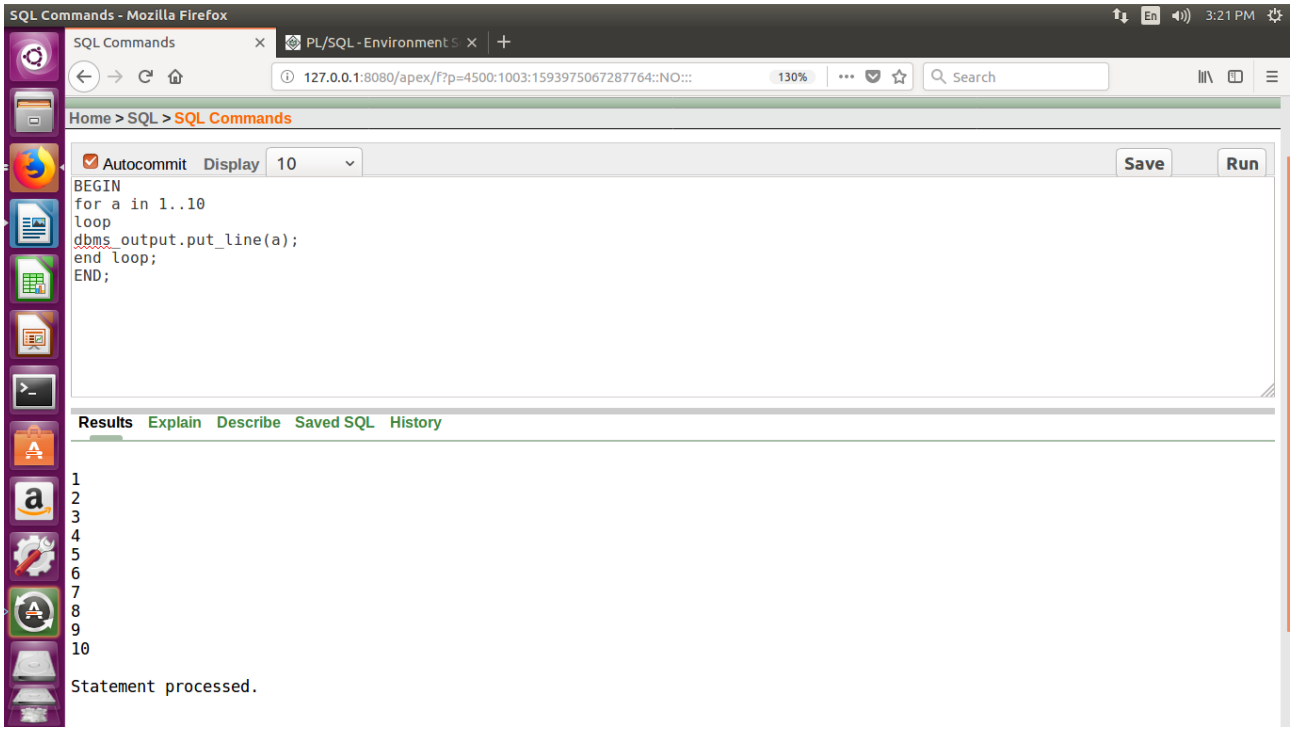

SQL Commands - Mozilla Firefox
2:56 PM
SQL Commands
PL/SQL - Environment
127.0.0.1:8080/apex/f?p=4500:1003:1593975067287764::NO:::
130%
Search
Home > SQL > SQL Commands
Autocommit
Display
10
Save
Run
BEGIN
insert into patient (pat_name, pat_id,contact, age)
values('AAA', '1000', '2500034','56');
end;
Results
Explain
Describe
Saved SQL
History
PAT_ID
PAT_NAME
CONTACT
AGE
123
sdfva
6757456
56
123
sdfva
6757456
56
444
ddd
6354634
62
456
d45
fhd45
23
1000
AAA
2500034
56
5 rows returned in 0.00 seconds
CSV Export
Application Express 2.1.0.00.39
Language: en-us
Copyright © 1999, 2006, Oracle. All rights reserved.


SQL Commands - Mozilla Firefox
2:58 PM
SQL Commands
PL/SQL - Environment S
127.0.0.1:8080/apex/f?p=4500:1003:1593975067287764::NO:::
130%
Search
Home
Logout
Help
User: SYSTEM
Home > SQL > SQL Commands
Autocommit
Display
10
Save
Run
BEGIN
update patient set age='10' where pat_name='AAA';
end;
Results
Explain
Describe
Saved SQL
History
PAT_ID
PAT_NAME
CONTACT
AGE
123
sdfva
6757456
56
123
sdfva
6757456
56
444
ddd
6354634
62
456
d45
fhd45
23
1000
AAA
2500034
10
5 rows returned in 0.00 seconds
CSV Export
Application Express 2.1.0.00.39
Language: en-us
Copyright © 1999, 2006, Oracle. All rights reserved.


SQL Commands - Mozilla Firefox
2:59 PM
SQL Commands
PL/SQL - Environment S
127.0.0.1:8080/apex/f?p=4500:1003:1593975067287764::NO:::
130%
Search
Home
Logout
Help
User: SYSTEM
Home > SQL > SQL Commands
Autocommit
Display
10
Save
Run
BEGIN
delete patient where pat_id='123';
end;
Results
Explain
Describe
Saved SQL
History
PAT_ID
PAT_NAME
CONTACT
AGE
444
ddd
6354634
62
456
d45
fhd45
23
1000
AAA
2500034
10
3 rows returned in 0.00 seconds
CSV Export
Application Express 2.1.0.00.39
Language: en-us
Copyright © 1999, 2006, Oracle. All rights reserved.


SQL Commands - Mozilla Firefox
3:15 PM
SQL Commands
PL/SQL - Environment
127.0.0.1:8080/apex/f?p=4500:1003:1593975067287764::NO:::
130%
Search
ORACLE Database Express Edition
Home
Logout
Help
User: SYSTEM
Home > SQL > SQL Commands
Autocommit
Display
10
Save
Run
DECLARE
a integer;
b integer;
avg integer;
BEGIN
a:=:a;
b:=:b;
dbms_output.put_line('a='||a);
dbms_output.put_line('b='||b);
END
Results
Explain
Describe
Saved SQL
History
a=10
b=15
Statement processed.
0.00 seconds
Application Express 2.1.0.00.39
Language: en-us
Copyright © 1999, 2006, Oracle. All rights reserved.


SQL Commands - Mozilla Firefox
3:19 PM
SQL Commands
PL/SQL - Environment
127.0.0.1:8080/apex/f?p=4500:1003:1593975067287764::NO:::
130%
Search
ORACLE Database Express Edition
Home
Logout
Help
User: SYSTEM
Home > SQL > SQL Commands
Autocommit
Display
10
Save
Run
DECLARE
a integer;
b integer;
BEGIN
a:=:a;
b:=:b;
dbms_output.put_line('a='||a);
dbms_output.put_line('b='||b);
dbms_output.put_line('sum='||(a+b));
dbms_output.put_line('avg='||((a+b)/2));
END
Results
Explain
Describe
Saved SQL
History
a=10
b=15
sum=25
avg=12.5
Statement processed.
0.00 seconds


SQL Commands - Mozilla Firefox
3:21 PM
SQL Commands
PL/SQL - Environment S
127.0.0.1:8080/apex/f?p=4500:1003:15939750672877 64::NO:::
130%
Search
Home > SQL > SQL Commands
Autocommit
Display
10
Save
Run
BEGIN
for a in 1..10
loop
dbms_output.put_line(a);
end loop;
END;
Results
Explain
Describe
Saved SQL
History
1
2
3
4
5
6
7
8
9
10
Statement processed.