GROUP A Assignment - 1

Name : Pavan Patil
Rollno. : 51

TITLE : Study and design a database with suitable example using following database systems:
 Relational: SQL / PostgreSQL / MySQL
 Key-value: Riak / Redis
 Columnar: Hbase
 Document: MongoDB / CouchDB
 Graph: Neo4J
Compare the different database systems based on points like efficiency, scalability, characteristics and performance.

Study the SQLite database and its uses. Also elaborate on building and installing of SQLite.

THEORY:

Relational databases :

SQL :

1. SQL stands for Structured Query Language. It is used for storing and managing data in relational database management system (RDMS).
2. It is a standard language for Relational Database System. It enables a user to create, read, update and delete relational databases and tables.
3. All the RDBMS like MySQL, Informix, Oracle, MS Access and SQL Server use SQL as their standard database language.
4. SQL allows users to query the database in a number of ways, using English-like statements.

PostgreSQL :

What is PostgreSQL?
PostgreSQL (pronounced as **post-gress-Q-L**) is an open source relational database management system (DBMS) developed by a worldwide team of volunteers. PostgreSQL is not controlled by any corporation or other private entity and the source code is available free of charge.

Key Features of PostgreSQL

PostgreSQL runs on all major operating systems, including Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), and Windows. It supports text, images, sounds, and video, and includes programming interfaces for C / C++, Java, Perl, Python, Ruby, Tcl and Open Database Connectivity (ODBC).

PostgreSQL supports a large part of the SQL standard and offers many modern features including the following −

5. Complex SQL queries
6. SQL Sub-selects
7. Foreign keys
8. Trigger
9. Views
10. Transactions
11. Multiversion concurrency control (MVCC)
12. Streaming Replication (as of 9.0)
13. Hot Standby (as of 9.0)

You can check official documentation of PostgreSQL to understand the above-mentioned features. PostgreSQL can be extended by the user in many ways. For example by adding new −

1) Data types
2) Functions
3) Operators
4) Aggregate functions
5) Index methods

MySQL :

MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.

1. MySQL uses a standard form of the well-known SQL data language.
2. MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
3. MySQL works very quickly and works well even with large data sets.
4. MySQL is very friendly to PHP, the most appreciated language for web development.
5. MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).

6. MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

Key value :

Riak :

Riak is a distributed NoSQL key-value data store that offers high availability, fault tolerance, operational simplicity, and scalability. ... Written in Erlang, **Riak** has fault tolerant data replication and automatic data distribution across the cluster for performance and resilience.

Redis :

Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker. It supports data structures such
as strings, hashes, lists, sets, sorted sets with range
queries, bitmaps, hyperloglogs, geospatial indexes with radius queries and streams. Redis has built-in replication, Lua scripting, LRU eviction, transactions and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster.

Columnar :

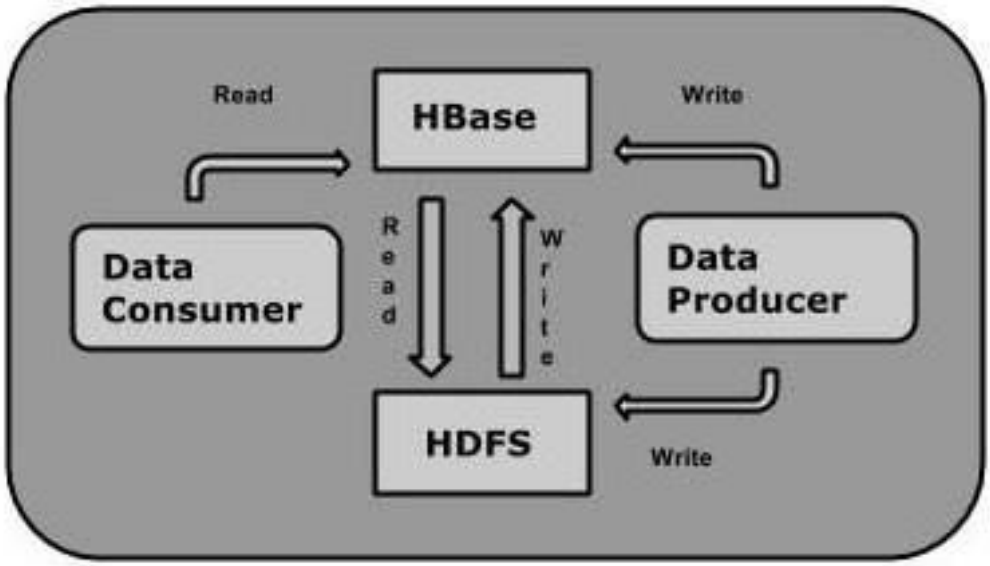
Hbase :

HBase is a distributed column-oriented database built on top of the Hadoop file system. It is an open-source project and is horizontally scalable.

HBase is a data model that is similar to Google's big table designed to provide quick random access to huge amounts of structured data. It leverages the fault tolerance provided by the Hadoop File System (HDFS).

It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System.

One can store the data in HDFS either directly or through HBase. Data consumer reads/accesses the data in HDFS randomly using HBase. HBase sits on top of the Hadoop File System and provides read and write access.



Document :

MongoDB :

MongoDB is a NoSQL database which stores the data in form of key-value pairs. It is an **Open Source**, **Document Database** which provides high performance and scalability along with data modelling and data management of huge sets of data in an enterprise application.

These are some important features of MongoDB:

1. Support ad hoc queries

In MongoDB, you can search by field, range query and it also supports regular expression searches.

Indexing

You can index any field in a document.

3. Replication

MongoDB supports Master Slave replication.

A master can perform Reads and Writes and a Slave copies data from the master and can only be used for reads or back up (not writes)

4. Duplication of data

MongoDB can run over multiple servers. The data is duplicated to keep the system up and also keep its running condition in case of hardware failure.

5. Load balancing

It has an automatic load balancing configuration because of data placed in shards.

6. Supports map reduce and aggregation tools.

7. Uses JavaScript instead of Procedures.

8. It is a schema-less database written in C++.

9. Provides high performance.

CouchDB :

CouchDB is a document storage NoSQL database. It provides the facility of storing documents with unique names, and it also provides an API called RESTful HTTP API for reading and updating (add, edit, delete) database documents. ... The database will not have any partially saved or edited documents.

Features of CouchDB

Following is a list of most attractive features of CouchDB:

Document Storage: CouchDB is a NoSQL database which follows document storage. Documents are the primary unit of data where each field is uniquely named and contains values of various data types such as text, number, Boolean, lists, etc.

Documents don't have a set limit to text size or element count.

Browser Based GUI: CouchDB provides an interface Futon which facilitates a browser based GUI to handle your data, permission and configuration.

Replication: CouchDB provides the simplest form of replication. There is no other database is so simple to replicate.

ACID Properties: The CouchDB file layout follows all the features of ACID properties. Once the data is entered in to the disc, it will not be overwritten. Document updates (add, edit, delete) follow Atomicity, i.e., they will be saved completely or not saved at all. The database will not

have any partially saved or edited documents. Almost all of these update are serialized and any number of clients can read a document without waiting and without being interrupted.

JSONP for Free: If you update your config to allow_jsonp = true then your database is accessible cross domain for GET requests.

Authentication and Session Support: CouchDB facilitates you to keep authentication open via a session cookie like web application.

Security: CouchDB also provides database-level security. The permissions per database are separated into readers and admins. Readers can both read and write to the database.

Validation: You can validate the inserted data into the database by combining with authentication to ensure the creator of the document is the one who is logged in.

Map/Reduce List and Show: The main reason behind the popularity of MongoDB and CouchDB is map/reduce system.

Graph :

Neo4J :

Neo4j structure:

Neo4j stores and present the data in the form of graph not in tabular format or not in a Jason format. Here the whole data is represented by nodes and there you can create a relationship between nodes. That means the whole database collection will look like a graph, that's why it is making it unique from other database management system. MS Access, SQL server all the relational database management system use tables to store or present the data with the help of column and row but Neo4j doesn't use tables, row or columns like old school style to store or present the data.

Neo4j Usage:

If your Database Management System has so many interconnecting relationships then you can use Neo4j that will be the best choice. Neo4j is highly preferable to store data that contains multiple connections between nodes. This is where the Neo4j(Graph Database) comes in it's more comfortable to use with relational data than the relational database. Because Neo4j doesn't require a predefined schema, you just need to load the data here the data is the main structure. It is schema optional Database Management System.

There are some unique features that will make you chose Neo4j over any other Database Management System. Neo4j is surrounded by relationships but there is no need to set up primary key or foreign key constraints to any data. Here you can add any relation between any nodes you want. That makes the Neo4j extremely suited for Networking data, below is the list of data areas where you can use this Database Management System.

- Social network like in Facebook, Twitter or in Instagram

- Network Diagram
- Fraud Detection
- Graph based searched of digital assets
- Data Management

The Main Differences between RDBMS and NoSQL:

1. **Type –**
   SQL databases are primarily called as Relational Databases (RDBMS); whereas NoSQL database are primarily called as non-relational or distributed database.
2. **Language –**
   SQL databases defines and manipulates data based structured query language (SQL). Seeing from a side this language is extremely powerful. SQL is one of the most versatile and widely-used options available which makes it a safe choice especially for great complex queries. But from other side it can be restrictive. SQL requires you to use predefined schemas to determine the structure of your data before you work with it. Also all of your data must follow the same structure. This can require significant up-front preparation which means that a change in the structure would be both difficult and disruptive to your whole system.

   A NoSQL database has dynamic schema for unstructured data. Data is stored in many ways which means it can be document-oriented, column-oriented, graph-based or organized as a KeyValue store. This flexibility means that documents can be created without having defined structure first. Also each document can have its own unique structure. The syntax varies from database to database, and you can add fields as you go.
3. **The Scalability –**
   In almost all situations SQL databases are vertically scalable. This means that you can increase the load on a single server by increasing things like RAM, CPU or SSD. But on the other hand NoSQL databases are horizontally scalable. This means that you handle more traffic by sharding, or adding more servers in your NoSQL database. It is similar to adding more floors to the same building versus adding more buildings to the neighborhood. Thus NoSQL can ultimately become larger and more powerful, making these databases the

preferred choice for large or ever-changing data sets.

4. **The Structure –**
   SQL databases are table-based on the other hand NoSQL databases are either key-value pairs, document-based, graph databases or wide-column stores. This makes relational SQL databases a better option for applications that require multi-row transactions such as an accounting system or for legacy systems that were built for a relational structure.
5. **Property followed –**
   SQL databases follow ACID properties (Atomicity, Consistency, Isolation and Durability) whereas the NoSQL database follows the Brewers CAP theorem (Consistency, Availability and Partition tolerance).
6. **Support –**
   Great support is available for all SQL database from their vendors. Also a lot of independent consultations are there who can help you with SQL database for a very large scale deployments but for some NoSQL database you still have to rely on community support and only limited outside experts are available for setting up and deploying your large scale NoSQL deployments.

   Some examples of SQL databases include PostgreSQL, MySQL, Oracle and Microsoft SQL Server. NoSQL database examples include Redis, RavenDB Cassandra, MongoDB, BigTable, HBase, Neo4j and CouchDB.

GROUP A Assignment - 3

Name : Pavan Patil
Rollno. : 51

TITLE : Study the SQLite database and its uses. Also elaborate on building and installing of SQLite.

THEORY:

What is SQLite?

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. It is a database, which is zero-configured, which means like other databases you do not need to configure it in your system.

SQLite engine is not a standalone process like other databases, you can link it statically or dynamically as per your requirement with your application. SQLite accesses its storage files directly.

Why SQLite?

- SQLite does not require a separate server process or system to operate (serverless).
- SQLite comes with zero-configuration, which means no setup or administration needed.
- A complete SQLite database is stored in a single cross-platform disk file.
- SQLite is very small and light weight, less than 400KiB fully configured or less than 250KiB with optional features omitted.
- SQLite is self-contained, which means no external dependencies.
- SQLite transactions are fully ACID-compliant, allowing safe access from multiple processes or threads.
- SQLite supports most of the query language features found in SQL92 (SQL2) standard.
- SQLite is written in ANSI-C and provides simple and easy-to-use API.
- SQLite is available on UNIX (Linux, Mac OS-X, Android, iOS) and Windows (Win32, WinCE, WinRT).

Package Installer

Installation packages available for Windows 10 users:

From the SQLite official website in the download section. The following screenshot allows you to download different SQLite's installation packages for Windows:

The command line shell program:

The highlighted download package is called the **Command-Line Program (CLP)**. CLP is a command line application that let you access the SQLite database management system and all the features of the SQLite. Using CLP, you can create and manage the SQLite database. And it is the tool that we will use throughout the tutorial.

- 32-bit DLL(x86): The SQLite Database system core library for x86 platforms.
- 64-bit DLL (x64): The SQLite Database system core library for x64 platforms.

Installing the Command-Line Program (CLP) on your machine:

In the following steps, you will find the steps for how to install the Command-Line Program (CLP) on your machine:

Step 1) Download the highlighted download package from the previous image to your PC. It is a "**zip**" file.

Step 2) Extract the zip file. You will find the "**sqlite3.exe**" in the extracted file as following:

Step 3) Open My Computer, and double-click partition **"C"** to navigate to it:

Step 4) Create a new directory "**sqlite**":

Step 5) Copy the file "**sqlite3.exe**" into it. This is what we will use through the tutorials to run SQLite queries:

Building Sample database

In the following steps, we will create the sample database that we will use throughout the tutorials:

Step 1) Open a text file and paste the following commands into it:

```
CREATE TABLE [Departments] (
   [DepartmentId] INTEGER  NOT NULL PRIMARY KEY,
   [DepartmentName] NVARCHAR(50)  NULL
);
INSERT INTO Departments VALUES(1, 'IT');
INSERT INTO Departments VALUES(2, 'Physics');
INSERT INTO Departments VALUES(3, 'Arts');
INSERT INTO Departments VALUES(4, 'Math');

CREATE TABLE [Students] (
   [StudentId] INTEGER  PRIMARY KEY NOT NULL,
   [StudentName] NVARCHAR(50) NOT NULL,
   [DepartmentId] INTEGER  NULL,
```

```
   [DateOfBirth] DATE NULL,
   FOREIGN KEY(DepartmentId) REFERENCES Departments(DepartmentId)
);
INSERT INTO Students VALUES(1, 'Michael', 1, '1998-10-12');
INSERT INTO Students VALUES(2, 'John', 1, '1998-10-12');
INSERT INTO Students VALUES(3, 'Jack', 1, '1998-10-12');
INSERT INTO Students VALUES(4, 'Sara', 2, '1998-10-12');
INSERT INTO Students VALUES(5, 'Sally', 2, '1998-10-12');
INSERT INTO Students VALUES(6, 'Jena', NULL, '1998-10-12');
INSERT INTO Students VALUES(7, 'Nancy', 2, '1998-10-12');
INSERT INTO Students VALUES(8, 'Adam', 3, '1998-10-12');
INSERT INTO Students VALUES(9, 'Stevens', 3, '1998-10-12');
INSERT INTO Students VALUES(10, 'George', NULL, '1998-10-12');

CREATE TABLE [Tests] (
   [TestId] INTEGER NOT NULL PRIMARY KEY,
   [TestName] NVARCHAR(50) NOT NULL,
   [TestDate] DATE NULL
);
INSERT INTO [Tests] VALUES(1, 'Mid Term IT Exam', '2015-10-18');
INSERT INTO [Tests] VALUES(2, 'Mid Term Physics Exam', '2015-10-23');
INSERT INTO [Tests] VALUES(3, 'Mid Term Arts Exam', '2015-10-10');
INSERT INTO [Tests] VALUES(4, 'Mid Term Math Exam', '2015-10-15');

CREATE TABLE [Marks] (
   [MarkId] INTEGER NOT NULL PRIMARY KEY,
   [TestId] INTEGER NOT NULL,
   [StudentId] INTEGER  NOT NULL,
   [Mark] INTEGER  NULL,
   FOREIGN KEY(StudentId) REFERENCES Students(StudentId),
   FOREIGN KEY(TestId) REFERENCES Tests(TestId)
);

INSERT INTO Marks VALUES(1, 1, 1, 18);
INSERT INTO Marks VALUES(2, 1, 2, 20);
INSERT INTO Marks VALUES(3, 1, 3, 16);
INSERT INTO Marks VALUES(4, 2, 4, 19);
INSERT INTO Marks VALUES(5, 2, 5, 14);
INSERT INTO Marks VALUES(6, 2, 7, 20);
INSERT INTO Marks VALUES(7, 3, 8, 20);
INSERT INTO Marks VALUES(8, 3, 9, 20);
```

Step 2) Save the file as **"TutorialsSampleDB.sql"** in the following directory "**C:\sqlite**".

Step 3) Open the Windows Command Line tool (cmd.exe) from the start menu, type **"cmd"** and open it.

Step 4) It will open in the default path, you need to navigate to the **"C:\sqlite"** folder we had created earlier in this tutorial by the following command **"cd "C:\sqlite":**

Step 5) Write the following command,

```
sqlite3 TutorialsSampleDB.db < TutorialsSampleDB.sql
```

The command should be completed successfully, and you should see no output after that command as the following screenshot:

Step 6) You should now be able to see the database file "**TutorialsSampleDB.db**" created in the directory "**C:\sqlite**":

SQLite Create table

Syntax

Below is the syntax of CREATE TABLE statement.

```
CREATE TABLE table_name(
column1 datatype,
column1 datatype
);
```

To create a table, you should use the **"CREATE TABLE"** Query as follows:

```
CREATE TABLE guru99 (
  Id Int,
  Name Varchar
);
```

Drop table

To drop a table, use the **"DROP TABLE"** command followed by the table name as follows:

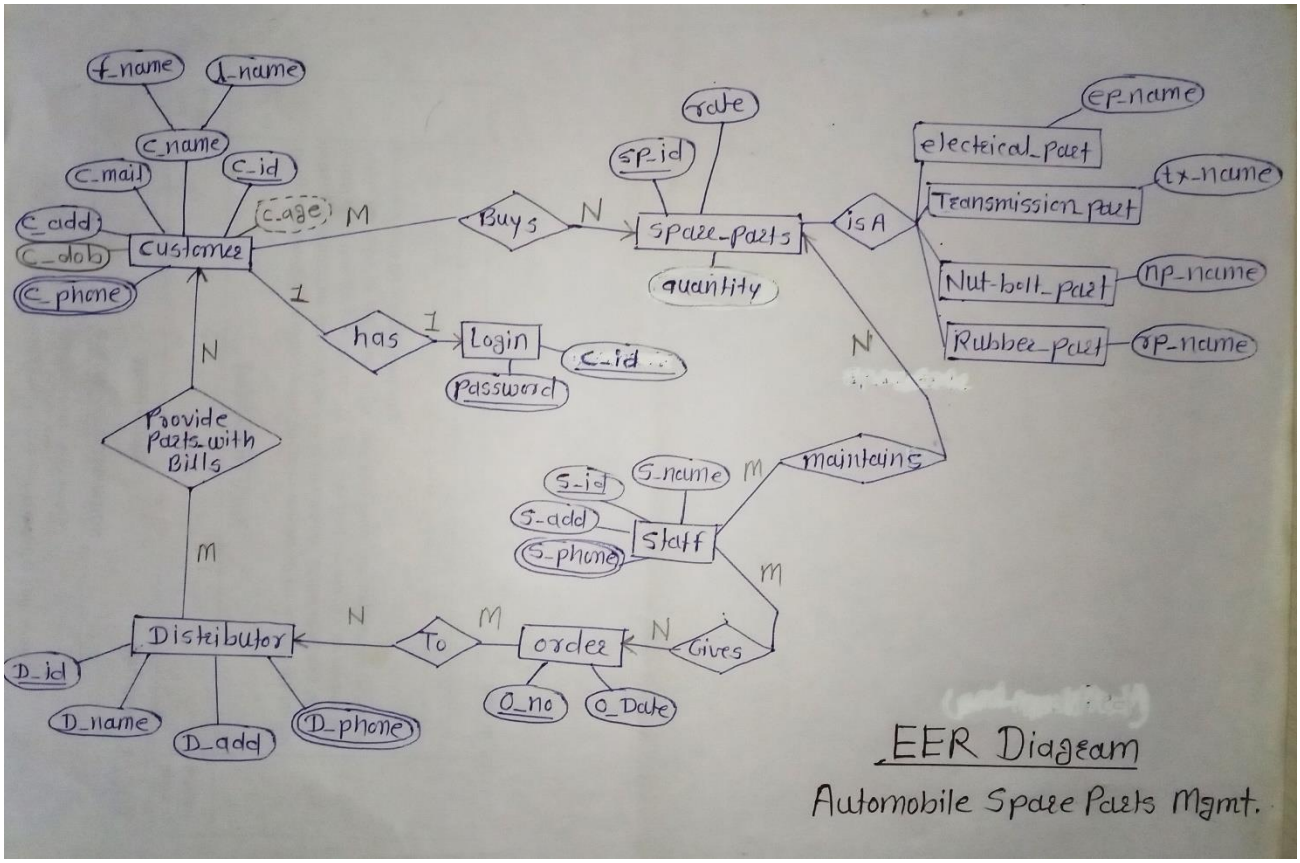
```
DROP TABLE guru99;
```

GROUP B Assignment - 1

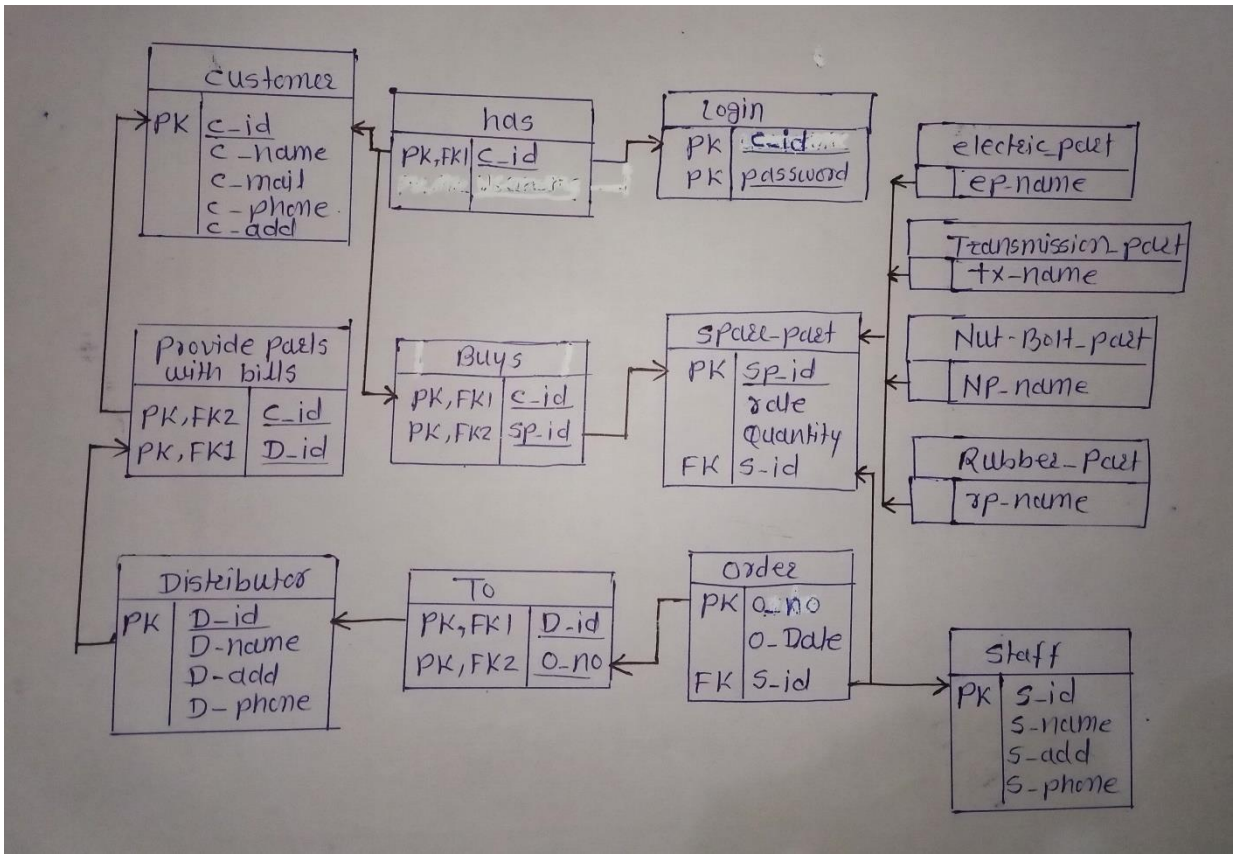
Name : Pavan Patil
Rollno. : 51

1. Design any database with at least 3 entities and relationships between them. Apply DCL and DDL commands. Draw suitable ER/EER diagram for the system.

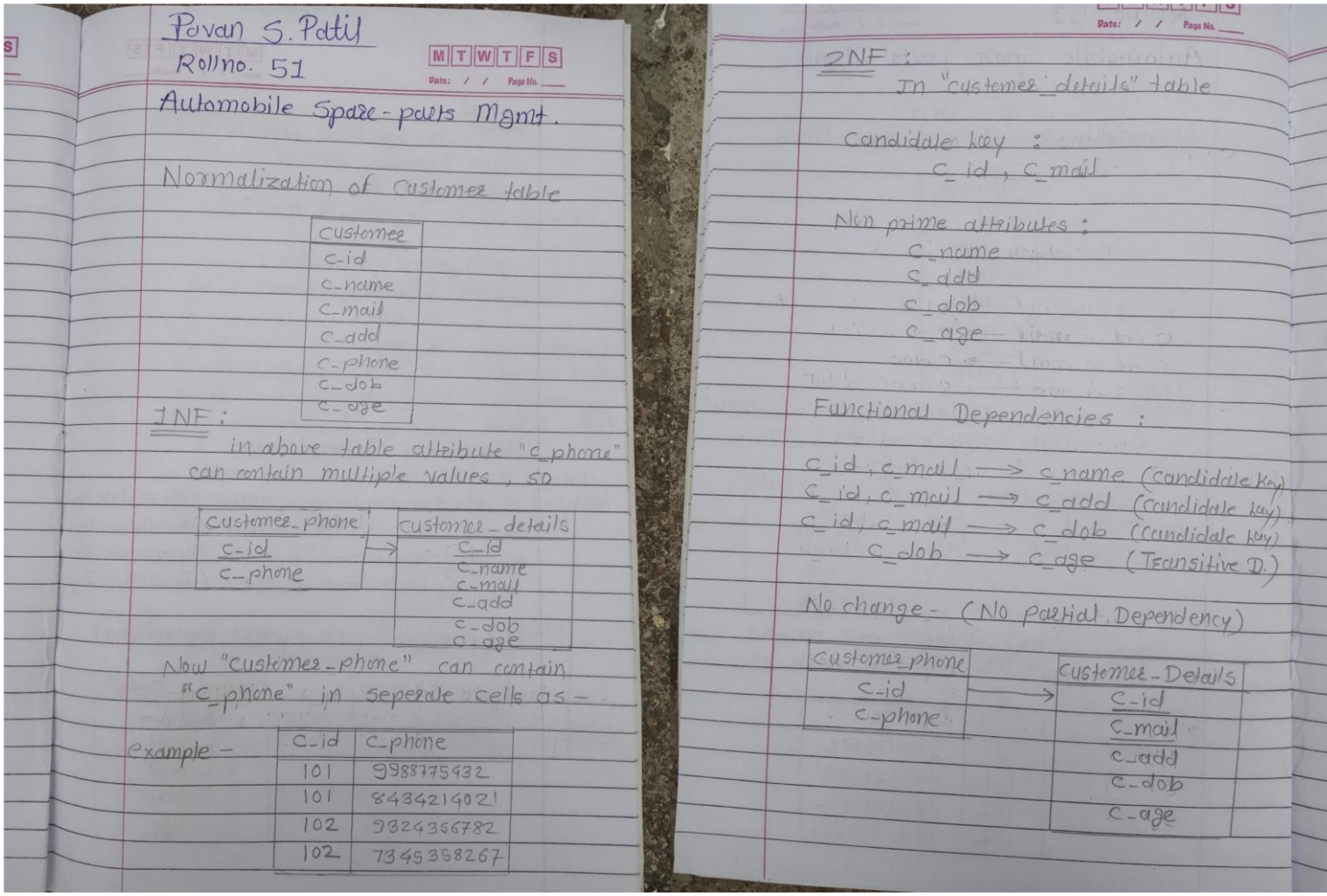
EER Diagram (Automobile Spare parts Management) –



Relational Model –



Normalization -



Pavan S. Patil
Roll no. 51

Automobile Spare-parts Mgmt.

Normalization of customer table

customer
c_id
c_name
c_mail
c_add
c_phone
c_dob
c_age

1NF :
in above table attribute "c_phone" can contain multiple values, so

customer_phone	customer_details
c_id	c_id
c_phone	c_name
	c_mail
	c_add
	c_dob
	c_age

Now "customer_phone" can contain "c_phone" in seperate cells as -

Example -

c_id	c_phone
101	9988775432
101	8434214021
102	9324356782
102	7345358267

2NF :
In "customer_details" table

Candidate key :
c_id, c_mail

Non prime attributes :
c_name
c_add
c_dob
c_age

Functional Dependencies :

c_id, c_mail → c_name (candidate key)
c_id, c_mail → c_add (candidate key)
c_id, c_mail → c_dob (candidate key)
c_dob → c_age (Transitive D.)

No change - (No partial Dependency)

customer_phone	customer_Details
c_id	c_id
c_phone	c_mail
	c_add
	c_dob
	c_age

Date: / / Page No.

3NF :

In "customer_phone" Table

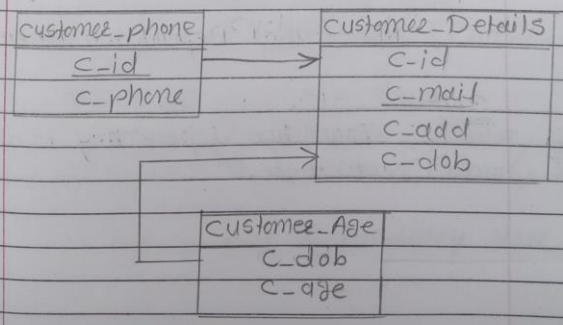
c_id → c_phone (Primary key)

In "customer_details" Table

c_id, c_mail → c_name (candidate K)
c_id, c_mail → c_add (candidate key)
c_id, c_mail → c_dob (candidate K.)
c_dob → c_age (Transitive D)

Here c_age is dependent on c_dob & c_dob is dependent on candidate key c_id, c_mail.

So,



M T W T F S
Date: / / Page No.

Normalization of "Spare_parts" Table.

Spare_parts
sp_id
rate
quantity

1NF: Here is No multi-valued attribute in this table. So No change.

2NF:
FD's —

sp_id → rate (Primary key)
sp_id → quantity (primary key)

No partial Dependency Here.

3NF :
Transitive dependency is also Not there.

Normalization of "Distributor" Table

Distributor
D-id
D-name
D-add
D-phone

1NF: S_phone can have multiple values, so

Distributor phone	→	Distributor_det
D-id		D-id
D-phone		D-name
		D-add

2NF: FD's –

In Distributor_phone Table
D_id → D_phone (Primary key)

In Distributor_Det Table
D_id → D_name (Primary key)
D_id → D_add (Primary key)

No partial Dependency Here

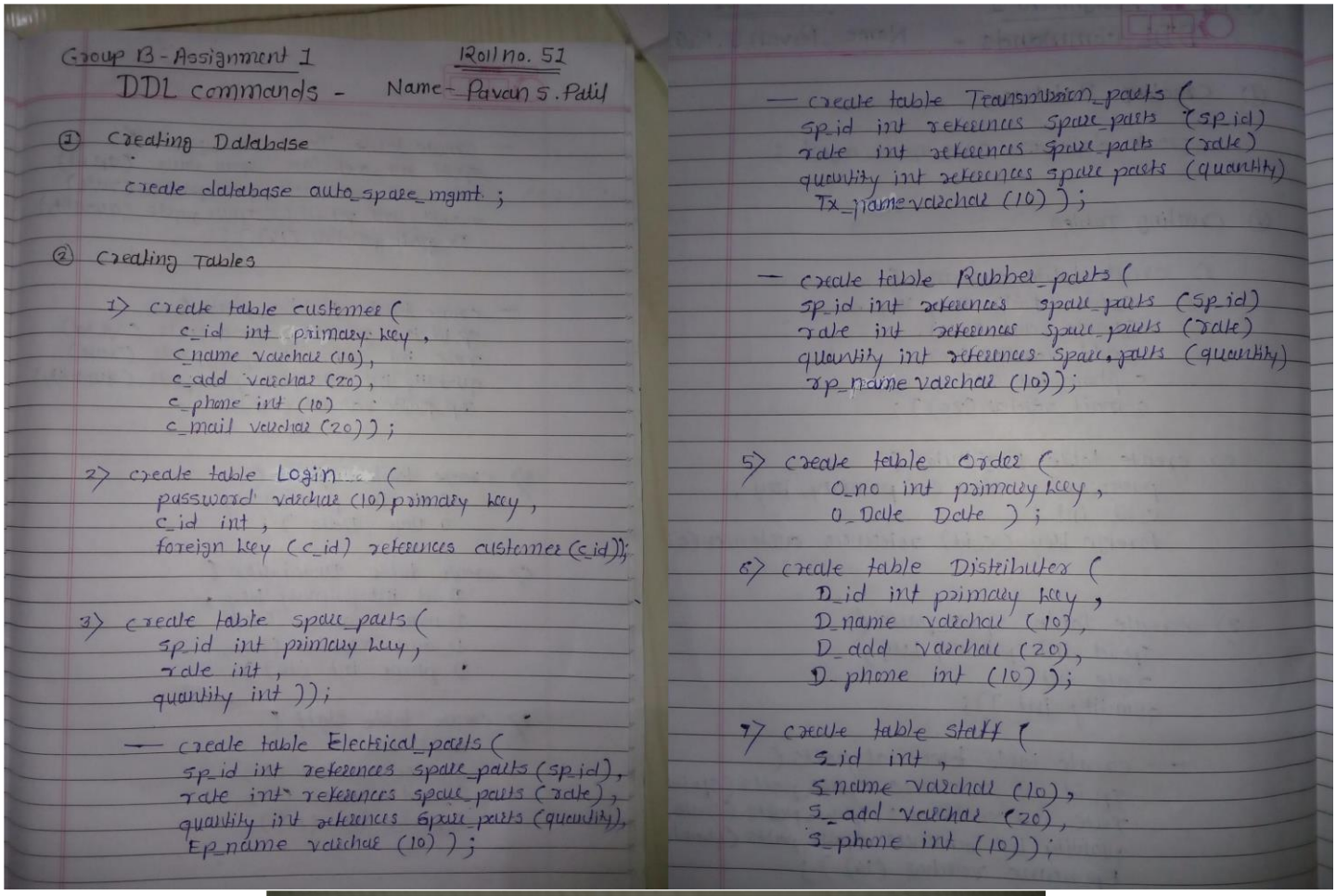
3NF: Transitive Dependency is also not there in Distributor Table.

* Similar for "Staff" Table for "S_phone" attribute.

* No need to Normalize order Table (because there is no multivalued attribute, No partial, Transitive Dependency)

O_no → O_Date

DDL, DCL Commands -



Group B - Assignment 1 Roll no. 51

DDL commands - Name - Pavan S. Patil

① Creating Database

```
create database auto_spare_mgmt ;
```

② Creating Tables

1) create table customer (
c_id int primary key ,
c_name varchar (10),
c_add varchar (20),
c_phone int (10)
c_mail varchar (20)) ;

2) create table Login ((
password varchar (10) primary key ,
c_id int ,
foreign key (c_id) references customer (c_id));

3) create table spare_parts (
sp_id int primary key,
rate int ,
quantity int));

— create table Electrical_parts (
sp_id int references spare_parts (sp_id),
rate int references spare_parts (rate),
quantity int references spare_parts (quantity),
Ep_name varchar (10)) ;

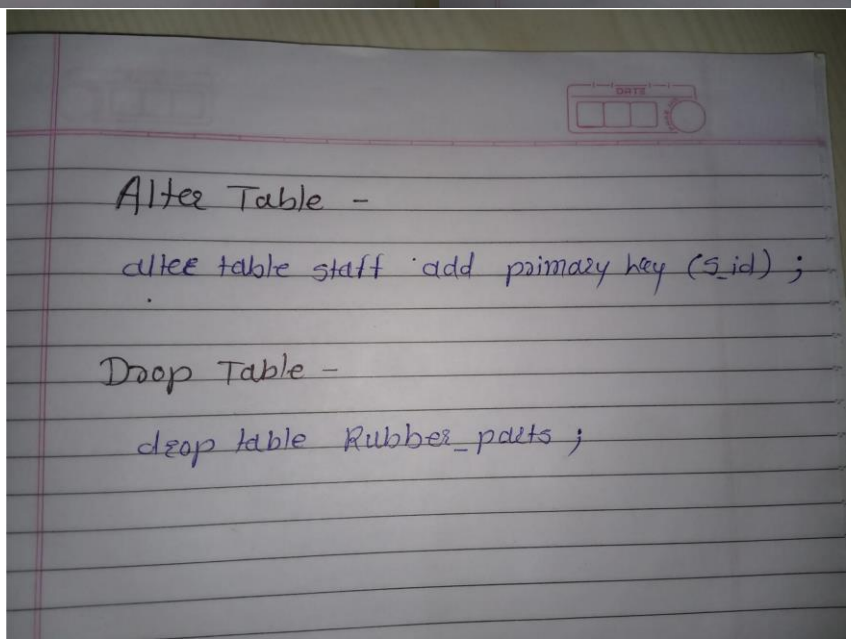
— create table Transmission_parts (
sp_id int references spare_parts (sp_id)
rate int references spare_parts (rate)
quantity int references spare_parts (quantity)
Tx_name varchar (10)) ;

— create table Rubber_parts (
sp_id int references spare_parts (sp_id)
rate int references spare_parts (rate)
quantity int references spare_parts (quantity)
rp_name varchar (10)) ;

5) create table Order (
O_no int primary key ,
O_Date Date) ;

6) create table Distributer (
D_id int primary key ,
D_name varchar (10),
D_add varchar (20),
D_phone int (10)) ;

7) create table staff (
S_id int ,
S_name varchar (10),
S_add varchar (20),
S_phone int (10)) ;



Alter Table -

```
altee table staff add primary key (S_id) ;
```

Droop Table -

```
droop table Rubber_parts ;
```

UOP Example 2 16/9/2020

Consider the relational database:
dept(dept_no, dname, loc, mgrcode)
emp(emp_no, ename, designation)
project(proj_no, proj_name,status)
dept and emp are related as 1 to many.
project and emp are related as 1 to many.
Write queries for the following:

1. List all employees of 'INVENTORY' department of 'PUNE' location.
2. Give the names of employees who are working on 'Blood Bank' project.
3. Given the name of managers from 'MARKETING' department.

Give all the employees working under status 'INCOMPLETE' projects

Queries –

```
create table dept(dept_no int primary key, dname varchar (10), loc
varchar (10), mgrcode int);

create table emp(emp_no int primary key, ename varchar (10),
designation varchar (10), dept_no int, proj_no int, foreign
key(dept_no) references dept(dept_no), foreign key (proj_no)
references project(proj_no));

create table project(proj_no int primary key, proj_name varchar
(10), status varchar (10)) ;

insert into dept values(11,"inventory", "pune",110);
insert into dept values(12,"management", "nashik",210);
insert into dept values(13,"marketing", "mumbai",310);
insert into dept values(14,"production", "delhi",410);

insert into project values(23,"blood bank", "complete");
insert into project values(35,"banking", "ready");
insert into project values(31,"hospital", "ongoing");
insert into project values(1,"hotel", "incomplete");


insert into emp values(310,"pavan", "manager",12,31);
insert into emp values(210,"vishal", "editor",14,23);
insert into emp values(110,"gaurav", "team",13,1);
insert into emp values(1,"abhi", "HR",11,35);
```

```
select *from dept;
select *from project;
select *from emp;

SELECT * FROM emp WHERE emp_no IN (SELECT emp_no FROM emp WHERE
dept_no IN (SELECT dept_no FROM dept WHERE loc="pune" AND
dname="inventory") );

SELECT ename FROM  emp WHERE proj_no IN (SELECT proj_no FROM project
WHERE proj_name = "blood bank");

SELECT ename FROM emp WHERE emp_no IN (SELECT mgrcode FROM dept
WHERE dname= "marketing");

SELECT * FROM emp WHERE emp_no IN (SELECT emp_no FROM emp WHERE
proj_no IN (SELECT proj_no FROM project WHERE status="incomplete")
);

create view v1 as (select emp_no, ename, designation from emp where
proj_no IN (select proj_no from project where status = "ongoing"));

delete from project where proj_no=1;

update v1 set ename="sagar" where emp_no=1;

insert into v1 values(220,"Aaditya", "staff");

select *from v1;
select *from emp;
```

Output –

```
dept_no dname   loc     mgrcode
11      inventory       pune    110
12      management      nashik  210
13      marketing       mumbai  310
14      production      delhi   410
proj_no proj_name       status
23      blood bank      complete
35      banking ready
31      hospital        ongoing
1       hotel   incomplete
emp_no  ename   designation     dept_no proj_no
310     pavan   manager 12      31
210     vishal  editor  14      23
110     gaurav  team    13      1
1       abhi    HR      11      35
emp_no  ename   designation     dept_no proj_no
1       abhi    HR      11      35
ename
vishal
ename
pavan
emp_no  ename   designation     dept_no proj_no
110     gaurav  team    13      1
emp_no  ename   designation
310     pavan   manager
emp_no  ename   designation     dept_no proj_no
310     pavan   manager 12      31
210     vishal  editor  14      23
110     gaurav  team    13      1
1       abhi    HR      11      35
220     Aaditya staff   NULL    NULL
```

UOP Example 3 17/9/2020

Consider following database:
Student(Roll_no, Name, Address)
Subject(sub_code, sub_name)
Write following queries in SQL

1. Find average marks of each student, along with the name of student.
2. Find how many students have failed in the subject “DBMS”.

(Note: fails menas obtained less than 40 marks)

Queries –

```
create table Student(roll_no integer primary key, name varchar (10),
address varchar(10));
create table Subject(sub_code integer primary key, sub_name varchar
(10));
create table Marks(roll_no integer,sub_code integer, marks
numeric(4,2), foreign key(roll_no) references
Student(roll_no),foreign key(sub_code) references
Subject(sub_code));

insert into Student values(1,"pavan","pimpri");
insert into Student values(2,"gaurav","nandura");
insert into Student values(3,"vishal","pune");
insert into Student values(4,"rutvik","mumbai");

insert into Subject values(111,"DBMS");
insert into Subject values(112,"TOC");
insert into Subject values(113,"OS");
insert into Subject values(114,"HCI");

select *from Student;
select *from Subject;

insert into Marks values(1,111,39);
insert into Marks values(1,112,30);
insert into Marks values(2,113,50);
insert into Marks values(2,114,33);
insert into Marks values(3,113,10);


select name, AVG(marks) from Marks, Student where Student.roll_no=
Marks.roll_no group by name;
```

```
select COUNT(*) FROM Marks, Subject where
Subject.sub_code=Marks.sub_code AND Subject.sub_name = "DBMS" AND
Marks.marks<40;
```

Output –

```
1       pavan   pimpri
2       gaurav  nandura
3       vishal  pune
4       rutvik  mumbai
111     DBMS
112     TOC
113     OS
114     HCI
pavan   34.500000
gaurav  41.500000
vishal  10.000000
1
```

University Database 21/9/2020

Schema –

```
create table student(
ID varchar(5), name varchar(20),
dept_name varchar(20),
tot_cred numeric(3,0),
primary key (ID),
foreign key (dept_name) references department(dept_name));


create table classroom(
building varchar(15),
room_number varchar(7),
capacity numeric(4,0),
primary key (building, room_number));


create table department(
dept_name varchar(20),
building varchar(15),
budget numeric(12,2),
primary key (dept_name));


create table course(
course_id varchar(8),
title varchar(50),
dept_name varchar(20),
credits numeric(2,0),
primary key (course_id),
foreign key (dept_name) references department(dept_name));


create table instructor(
ID varchar(5),
name varchar(20),
dept_name varchar(20),
salary numeric(8,2),
primary key (ID),
```

```
foreign key (dept_name) references department(dept_name));
```

```
create table section(
course_id varchar(8),
sec_id varchar(8),
semester varchar(6),
year numeric(4,0),
building varchar(15),
room_number varchar(7),
time_slot_id varchar(4),
primary key (course_id, sec_id, semester, year),
foreign key (course_id) references course(course_id),
foreign key (building, room_number) references
classroom(building,room_number));
```

```
create table teaches(
ID varchar(5),
course_id varchar(8),
sec_id varchar(8),
semester varchar(6),
year numeric(4,0),
primary key (ID, course_id, sec_id, semester, year),
foreign key (course_id,sec_id, semester, year) references
section(course_id,sec_id, semester, year),
foreign key (ID) references instructor(ID));
```

```
create table takes(
ID varchar(5),
course_id varchar(8),
sec_id varchar(8),
semester varchar(6),
year numeric(4,0),
grade varchar(2),
primary key (ID, course_id, sec_id, semester, year),
foreign key (course_id,sec_id, semester, year) references
section(course_id,sec_id,semester,year),
foreign key (ID) references student(ID));
```

```
create table advisor(
```

```
s_ID varchar(5),
i_ID varchar(5),
primary key (s_ID),
foreign key (i_ID) references instructor (ID),
foreign key (s_ID) references student(ID));


create table time_slot(
time_slot_id varchar(4),
day varchar(1),
start_hr numeric(2),
start_min numeric(2),
end_hr numeric(2),
end_min numeric(2),primary key (time_slot_id, day, start_hr,
start_min));


create table prereq(
course_id varchar(8),
prereq_id varchar(8),
primary key (course_id, prereq_id),
foreign key (course_id) references course(course_id),
foreign key (prereq_id) references course(course_id));



insert into classroom values ('Painter', '514', '10');
insert into classroom values ('Taylor', '3128', '70');
insert into classroom values ('Watson', '100', '30');
insert into classroom values ('Watson', '120', '50');


insert into department values ('Biology', 'Watson', '90000');
insert into department values ('Comp. Sci.', 'Taylor', '100000');
insert into department values ('Elec. Eng.', 'Taylor', '85000');
insert into department values ('Finance', 'Painter', '120000');


insert into course values ('BIO-301', 'Genetics', 'Biology', '4');
insert into course values ('BIO-399', 'Computational Biology',
'Biology', '3');
insert into course values ('CS-101', 'Intro. to Computer Science',
'Comp. Sci.', '4');
insert into course values ('CS-190', 'Game Design', 'Comp. Sci.',
'4');
```

```
insert into instructor values ('10101', 'James', 'Comp. Sci.',
'65000');
insert into instructor values ('12121', 'newton', 'Finance',
'90000');
insert into instructor values ('15151', 'Mozart', 'Music', null);
insert into instructor values ('22222', 'Einstein', 'Physics',
'95000');
insert into instructor values ('10105', 'adison', 'Comp. Sci.',
'90000');


insert into section values ('BIO-101', '1', 'Summer', '2009',
'Painter', '514', 'B');
insert into section values ('BIO-301', '1', 'Summer', '2010',
'Painter', '514', 'A');
insert into section values ('CS-101', '1', 'Fall', '2009', 'Packard',
'101', 'H');
insert into section values ('CS-101', '1', 'Spring', '2010',
'Packard', '101', 'F');
insert into section values ('CS-190', '1', 'Spring', '2009',
'Taylor', '3128', 'E');


insert into teaches values ('10101', 'CS-101', '1', 'Fall', '2009');
insert into teaches values ('10101', 'CS-315', '1', 'Spring',
'2010');
insert into teaches values ('10101', 'CS-347', '1', 'Fall', '2009');
insert into teaches values ('12121', 'FIN-201', '1', 'Spring',
'2010');
insert into teaches values ('15151', 'MU-199', '1', 'Spring',
'2010');
insert into teaches values ('22222', 'PHY-101', '1', 'Fall', '2009');


insert into student values ('00128', 'pavan', 'Comp. Sci.', '102');
insert into student values ('12345', 'vishal', 'Comp. Sci.', '32');
insert into student values ('19991', 'gaurav', 'History', '80');
insert into student values ('23121', 'Chavez', 'Finance', '110');


insert into takes values ('00128', 'CS-101', '1', 'Fall', '2009',
'A');
insert into takes values ('00128', 'CS-347', '1', 'Fall', '2009', 'A-
');
```

```
insert into takes values ('12345', 'CS-101', '1', 'Fall', '2009',
'C');
insert into takes values ('12345', 'CS-190', '2', 'Spring', '2009',
'A');
insert into takes values ('12345', 'CS-315', '1', 'Spring', '2010',
'A');


insert into advisor values ('00128', '45565');
insert into advisor values ('12345', '10101');
insert into advisor values ('23121', '76543');
insert into advisor values ('44553', '22222');
insert into advisor values ('45678', '22222');


insert into time_slot values ('A', 'F', '8', '0', '8', '50');
insert into time_slot values ('B', 'M', '9', '0', '9', '50');
insert into time_slot values ('B', 'W', '9', '0', '9', '50');
insert into time_slot values ('B', 'F', '9', '0', '9', '50');


insert into time_slot values ('C', 'F', '11', '0', '11', '50');
insert into time_slot values ('D', 'M', '13', '0', '13', '50');
insert into time_slot values ('E', 'T', '10', '30', '11', '45 ');
insert into time_slot values ('E', 'R', '10', '30', '11', '45 ');
insert into time_slot values ('F', 'T', '14', '30', '15', '45 ');
insert into time_slot values ('G', 'M', '16', '0', '16', '50');
insert into time_slot values ('G', 'W', '16', '0', '16', '50');


insert into prereq values ('BIO-301', 'BIO-101');
insert into prereq values ('BIO-399', 'BIO-101');
insert into prereq values ('CS-190', 'CS-101');
insert into prereq values ('CS-315', 'CS-101');
```

Operation Queries –

```
select ID, name, salary/12 from instructor;


/*select ID, name, salary/12  as salary;*/


select name from instructor where dept_name = "Comp. Sci.";


select name     from instructor  where dept_name = "Comp. Sci." and
salary > 80000;


select *from instructor, teaches;


select name, course_id from instructor , teaches where instructor.ID
= teaches.ID;


select name, course_id from instructor , teaches where instructor.ID
= teaches.ID  and  instructor. dept_name = "Music";


select distinct T.name from instructor as T, instructor as S where
T.salary > S.salary and S.dept_name = "Comp. Sci.";


select name     from instructor  where name like '%Ein%';


select distinct name from instructor order by name;


select name from instructor where salary between 80000 and 90000;


select name, course_id from instructor, teaches where (instructor.ID,
dept_name) = (teaches.ID, "Comp. Sci.");
```

```
(select course_id from section where semester = "Fall" and year =
2009) union (select course_id from section where semester = "Spring"
and year = 2010);


select distinct T.salary from instructor as T, instructor as S where
T.salary < S.salary;


select distinct salary from instructor;


select name     from instructor  where salary is null;


select avg (salary)from instructor where dept_name= "Comp. Sci.";


select count(distinct ID) from teaches where semester = "Spring" and
year = 2010;


select count(*)from course;


select dept_name, avg (salary)from instructor group by dept_name
having avg (salary) > 42000;


select sum(salary) from instructor;


select distinct course_id from section where semester = "Fall" and
year= 2009 and course_id in (select course_id from section where
semester = "Spring" and year= 2010);


select distinct course_id from section where semester = "Fall" and
year= 2009 and course_id  not in (select course_id from section where
semester = "Spring" and year= 2010);


select count(distinct ID) from takes where (course_id, sec_id,
semester, year) in (select course_id, sec_id, semester, year from
teaches where teaches.ID= 10101);
```

```
select distinct T.name from instructor as T, instructor as S where
T.salary > S.salary and S.dept_name = "Finance";


select name from instructor where salary > some (select salary from
instructor where dept_name = "Comp. Sci.");


select name from instructor where salary > all (select salary from
instructor where dept_name = "Finance");


select course_id from section as S where semester = "Fall" and year =
2009 and exists (select *from section as T where semester = "Spring"
and year= 2010 and S.course_id = T.course_id);


/*select distinct S.ID, S.name from student as S where not exists (
(select course_id from course where dept_name = "Physics")  except
(select T.course_id from takes as T where S.ID = T.ID));*/


select dept_name, avg_salary from (select dept_name, avg (salary)
from instructor group by dept_name) as dept_avg (dept_name,
avg_salary) where avg_salary > 42000;


with max_budget (value) as(select max(budget) from department) select
dept_name from department, max_budget where department.budget =
max_budget.value;


select dept_name,(select count(*)from instructor where
department.dept_name = instructor.dept_name)as num_instructors from
department;

delete from instructor where dept_name in (select dept_name from
department where building = "Watson");


/*delete from instructor where salary < (select avg(salary) from
instructor);*/
```

```
insert into course values ("CS-300", "Database Admin", "Comp. Sci.",
4);


insert into student select ID, name, dept_name, 0 from instructor;

update instructor set salary = salary * 1.03 where salary > 100000;
update instructor set salary = salary * 1.05 where salary <= 100000;


update instructor set salary = case when salary <= 100000 then salary
* 1.05 else salary * 1.03 end


/*update student S set tot_cred = (select sum(credits) from takes,
course where takes.course_id = course.course_id and S.ID=
takes.ID.and takes.grade <> 'F' and takes.grade is not null);*/
```

Output –

```
ID      name    salary/12
10101   James   5416.666667
12121   newton  7500.000000
15151   Mozart  NULL
22222   Einstein7916.666667
10105   adison  7500.000000


name
James
adison


name
adison


ID      name    dept_name       salary  ID      course_id       sec_id  semester
        year
10105   adison  Comp. Sci.      90000.00 10101  CS-101  1       Fall    2009
22222   Einstein Physics 95000.00 10101  CS-101  1       Fall    2009
15151   Mozart  Music   NULL    10101   CS-101  1       Fall    2009
12121   newton  Finance 90000.00 10101  CS-101  1       Fall    2009
10101   James   Comp. Sci.      65000.00 10101  CS-101  1       Fall    2009
10105   adison  Comp. Sci.      90000.00 10101  CS-315  1       Spring  2010
```

```
22222   Einstein Physics  95000.00 10101   CS-315  1       Spring  2010
15151   Mozart   Music    NULL     10101   CS-315  1       Spring  2010
12121   newton   Finance  90000.00 10101   CS-315  1       Spring  2010
10101   James    Comp. Sci.        65000.00 10101   CS-315  1       Spring  2010
10105   adison   Comp. Sci.        90000.00 10101   CS-347  1       Fall    2009
22222   Einstein Physics  95000.00 10101   CS-347  1       Fall    2009
15151   Mozart   Music    NULL     10101   CS-347  1       Fall    2009
12121   newton   Finance  90000.00 10101   CS-347  1       Fall    2009
10101   James    Comp. Sci.        65000.00 10101   CS-347  1       Fall    2009
10105   adison   Comp. Sci.        90000.00 12121   FIN-201 1       Spring  2010
22222   Einstein Physics  95000.00 12121   FIN-201 1       Spring  2010
15151   Mozart   Music    NULL     12121   FIN-201 1       Spring  2010
12121   newton   Finance  90000.00 12121   FIN-201 1       Spring  2010
10101   James    Comp. Sci.        65000.00 12121   FIN-201 1       Spring  2010
10105   adison   Comp. Sci.        90000.00 15151   MU-199  1       Spring  2010
22222   Einstein Physics  95000.00 15151   MU-199  1       Spring  2010
15151   Mozart   Music    NULL     15151   MU-199  1       Spring  2010
12121   newton   Finance  90000.00 15151   MU-199  1       Spring  2010
10101   James    Comp. Sci.        65000.00 15151   MU-199  1       Spring  2010
10105   adison   Comp. Sci.        90000.00 22222   PHY-101 1       Fall    2009
22222   Einstein Physics  95000.00 22222   PHY-101 1       Fall    2009
15151   Mozart   Music    NULL     22222   PHY-101 1       Fall    2009
12121   newton   Finance  90000.00 22222   PHY-101 1       Fall    2009
10101   James    Comp. Sci.        65000.00 22222   PHY-101 1       Fall    2009
```

```
name    course_id
James   CS-101
James   CS-315
James   CS-347
newton  FIN-201
Mozart  MU-199
Einstein PHY-101
```

```
name    course_id
Mozart  MU-199
```

```
name
newton
Einstein
adison
```

```
name
Einstein
```

```
name
adison
```

```
Einstein


James
Mozart
newton


name
newton
adison


name    course_id
James   CS-101
James   CS-315
James   CS-347


course_id
CS-101


salary
65000.00
90000.00







salary
65000.00
90000.00
NULL
95000.00


name
Mozart


avg (salary)
77500.000000


count(distinct ID)
3


count(*)
4
```

```
dept_name        avg (salary)
Comp. Sci.       77500.000000
Finance  90000.000000
Physics  95000.000000


sum(salary)
340000.00


course_id
CS-101


count(distinct ID)
2


name
Einstein


name
newton
Einstein
adison


name
Einstein


course_id
CS-101

dept_name        avg_salary
Comp. Sci.       77500.000000
Finance  90000.000000
Physics  95000.000000


dept_name
Finance


dept_name        num_instructors
Biology  0
Comp. Sci.       2
Elec. Eng.       0
Finance  1
```

Group B Assignment 3 24/9/2020

Schema –

```
create table Employee(emp_no int,name varchar(10),skill
varchar(20),pay_rate int,primary key(emp_no));


create table Position1(posting_no int, skill varchar(20),primary
key(posting_no));


create table Duty_allocation(posting_no int, emp_no int, day
varchar(10),shift varchar(5), foreign key(posting_no)references
Position1(posting_no), foreign key(emp_no) references
Employee(emp_no));


insert into Employee values(11, "gaurav", "coding", 40000);
insert into Employee values(13, "pavan", "testing", 30000);
insert into Employee values(12, "vishal", "marketing", 45000);
insert into Employee values(16, "rutvik", "presentation", 24000);



insert into Position1 values(23,"presentation");
insert into Position1 values(26,"coding");
insert into Position1 values(24,"testing");
insert into Position1 values(21,"marketing");



insert into Duty_allocation values(1123,13,"monday","day");
insert into Duty_allocation values(1121,11,"wednesday","night");
insert into Duty_allocation values(1124,16,"thursday","night");
insert into Duty_allocation values(1126,12,"saturday","day");
```

Operation Queries –

```
select posting_no, shift, day
from Duty_allocation, Employee
where Duty_allocation.emp_no = Employee.emp_no and
name = "vishal";
```

```
select shift, count(distinct emp_no)
from Duty_allocation
group by shift;
```

```
select Employee.emp_no, Position1.posting_no, Position1.skill
from Employee, Position1
where Employee.skill = Position1.skill;
```

```
SELECT emp_no, name
FROM Employee where pay_rate=(select MIN(pay_rate) from Employee);
```

```
select name, pay_rate
from Employee
where EXISTS
(select *
from Duty_allocation
where Employee.emp_no = Duty_allocation.emp_no);
```

```
SELECT COUNT(*) FROM Employee;
```

```
select sum(pay_rate) from Employee;
```

```
SELECT AVG(pay_rate), COUNT(*) FROM Employee;
```

```
SELECT MAX(pay_rate), MIN(pay_rate)
FROM Employee;
```

Output –

```
posting_no       shift    day
1126     day     Saturday



shift    count(distinct emp_no)
day      2
night    2



emp_no   posting_no       skill
16       23       presentation
11       26       coding
13       24       testing
12       21       marketing
emp_no   name
16       rutvik



name     pay_rate
gaurav   40000
vishal   45000
pavan    30000
rutvik   24000



COUNT(*)
4


sum(pay_rate)
139000


AVG(pay_rate)    COUNT(*)
34750.0000       4


MAX(pay_rate)    MIN(pay_rate)
45000    24000
```

Group B Assignment 4 28/9/2020

```
create table Project (project_id varchar(10) primary key , proj_name
varchar (30), chief_arch varchar(10));
create table Employee(emp_id int primary key , emp_name varchar
(10));


create table Assigned_To(project_id varchar(10) ,emp_id int, foreign
key (emp_id)references Employee(emp_id), foreign key
(project_id)references Project(project_id), Primary
key(project_id,emp_id) );


insert into Project values("C344","Bus Pass System", "gaurav");
insert into Project values("A892","Autospare management" ,"elliot");
insert into Project values("B672","Database Project" ,"rahul");
insert into Project values("C353"," Student Portal Project"
,"jemmy");


insert into Employee values(26, "jemmy");
insert into Employee values(19, "pavan");
insert into Employee values(23, "gaurav");
insert into Employee values(11, "rahul");
insert into Employee values(13, "ajay");
insert into Employee values(16, "rutvik");
insert into Employee values(17, "naval");

insert into Assigned_To values("B672", "11");
insert into Assigned_To values("C353", "26");
insert into Assigned_To values("C353", "19");
insert into Assigned_To values("A892", "23");


select emp_id
from Assigned_To
where project_id = "C353";


select emp_id,emp_name
from Assigned_To natural join Project natural join Employee
where project_id = "C353";
```

```
select emp_id,emp_name
from Assigned_To natural join Project natural join Employee
where proj_name = "Database Project";
```

```
select emp_id
from Employee
where emp_id not in
(select emp_id
from Assigned_To);
```

```
select project_id,proj_name,emp_id,emp_name from Employee natural
join Assigned_To natural join Project;
```

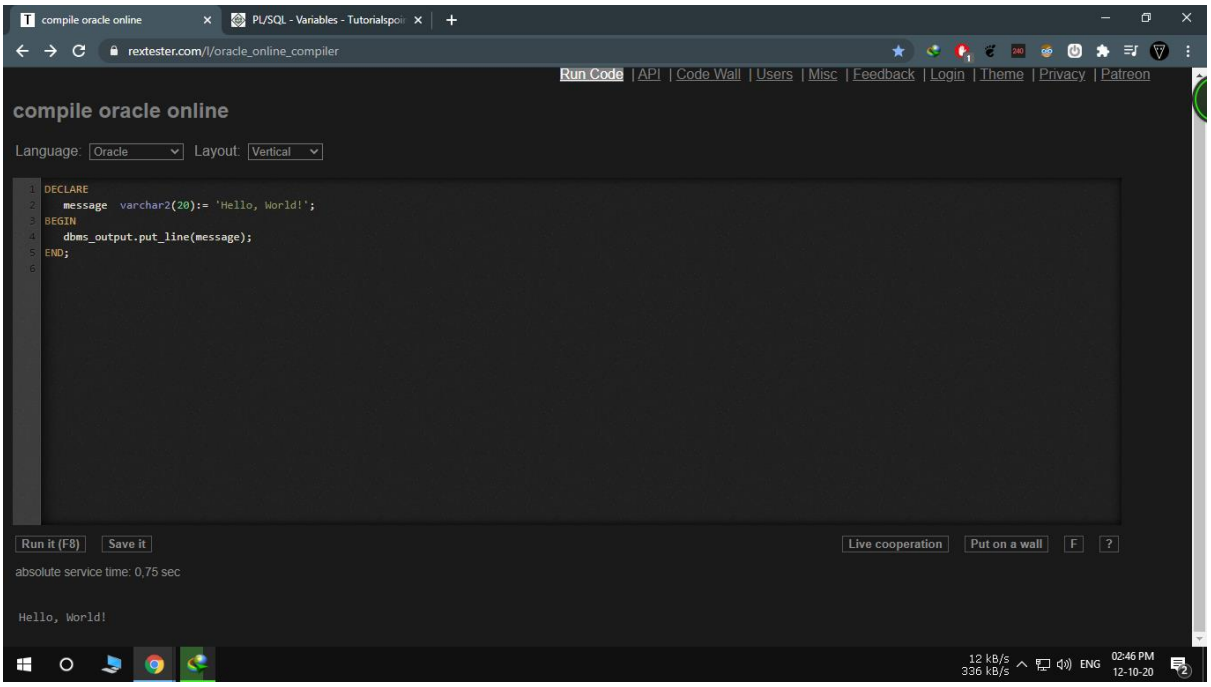
Output –

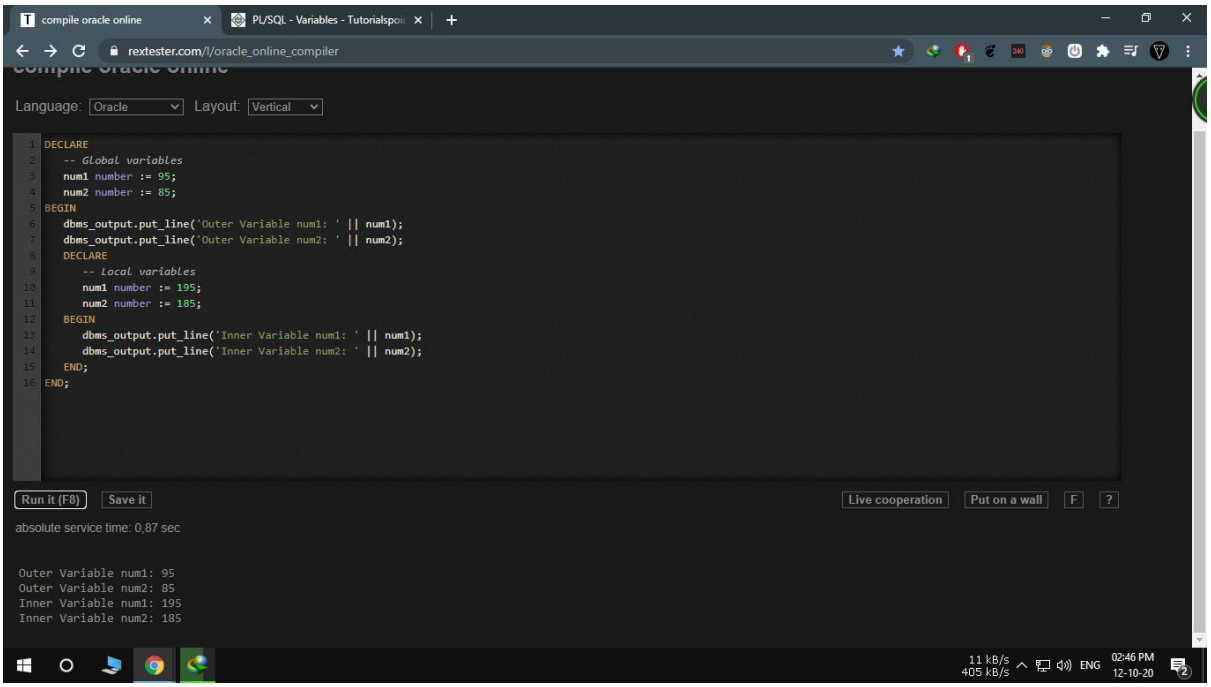
emp_id

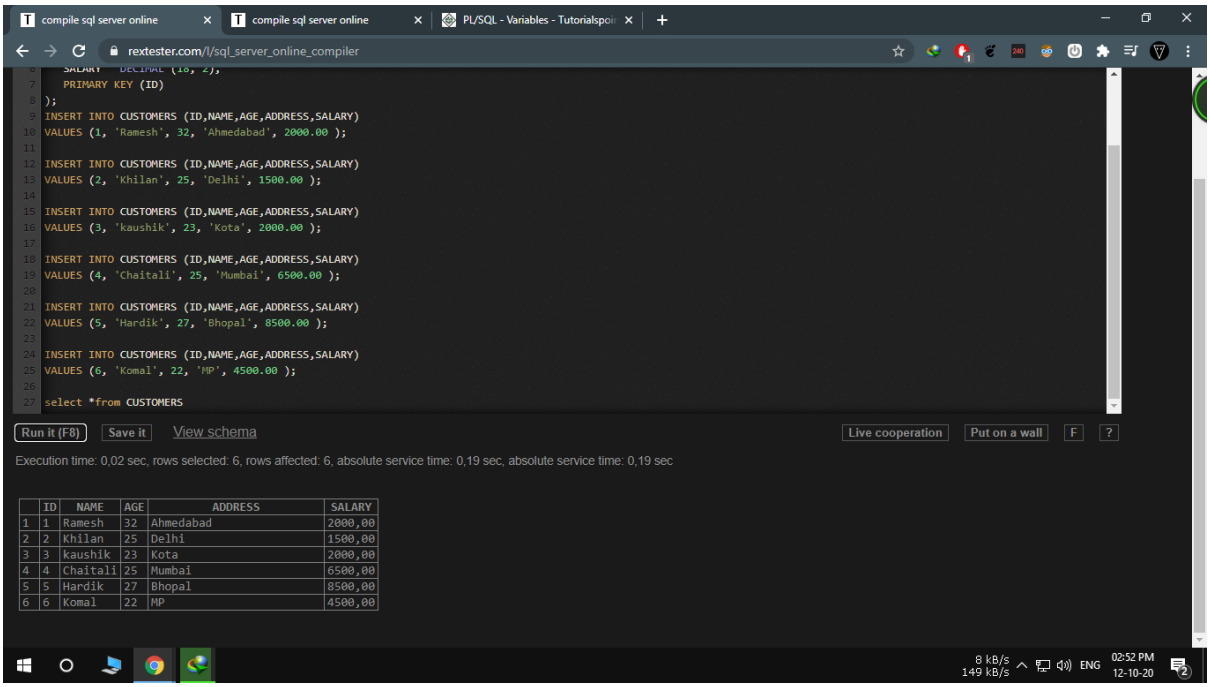
```
19
26
emp_id  emp_name
19      pavan
26      jemmy
emp_id  emp_name
11      rahul
emp_id
13
16
17
project_id      proj_name       emp_id  emp_name
A892    Autospare management    23      gaurav
B672    Database Project        11      rahul
C353    Student Portal Project 19       pavan
C353    Student Portal Project 26       jemmy
```

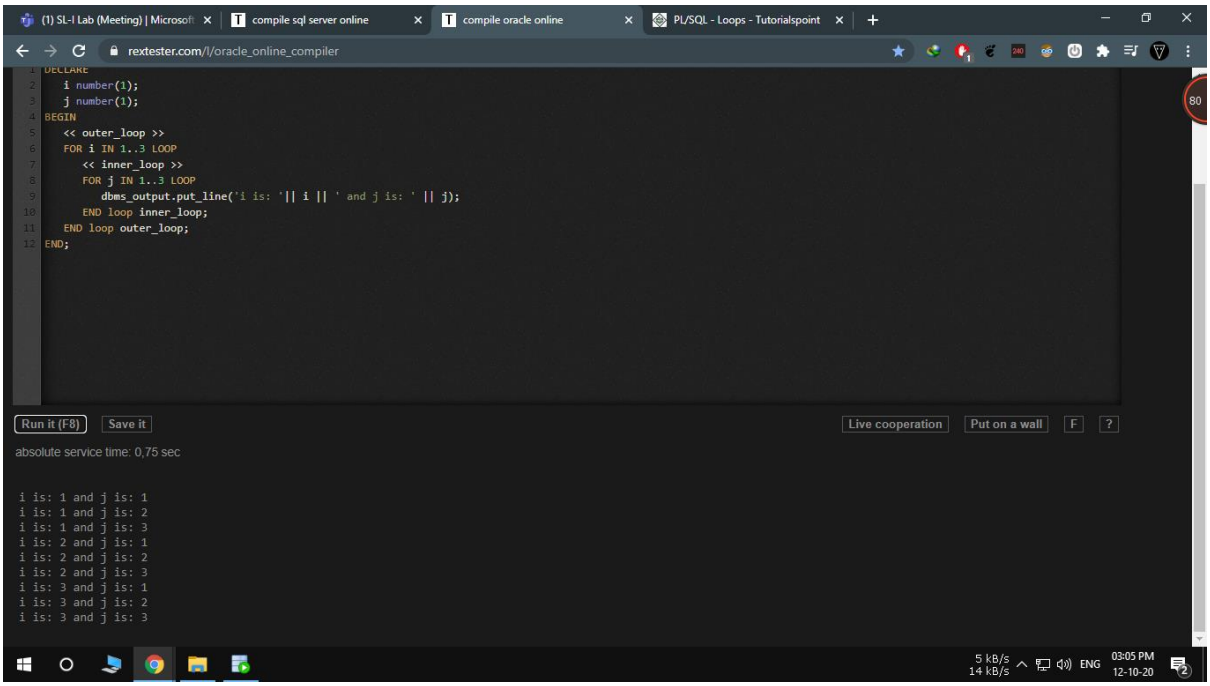
PL/SQL Examples 12/10/2020

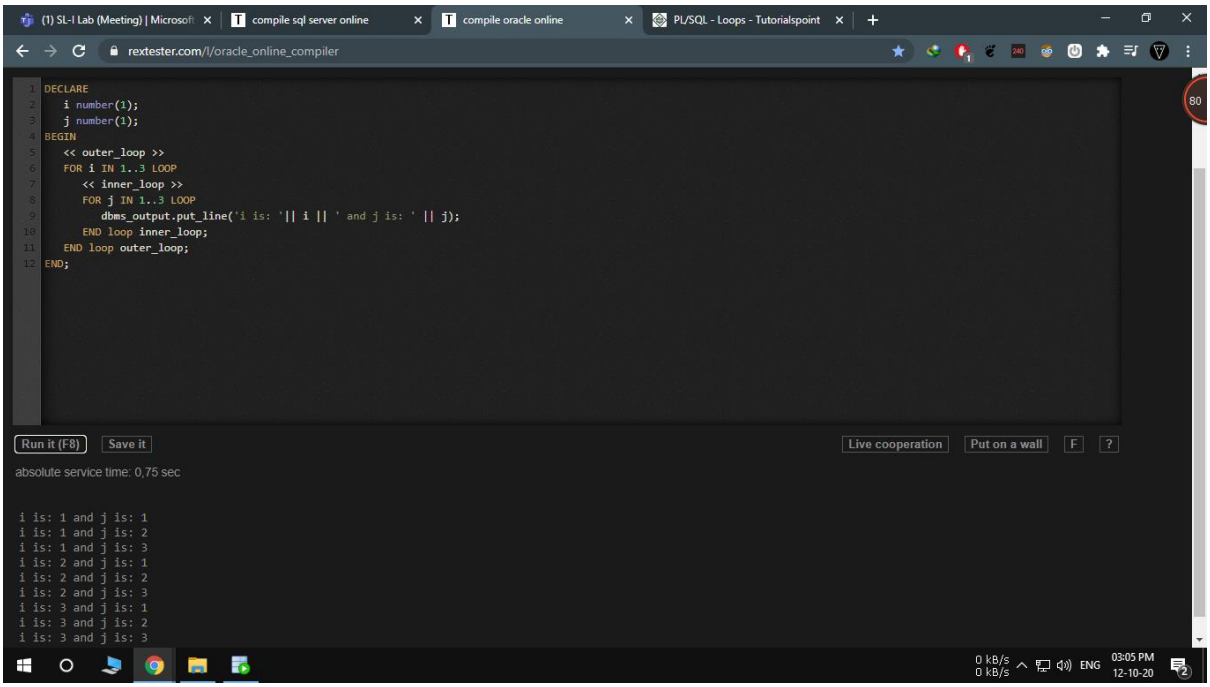
PAVAN S. PATIL
ROLLNO. 51

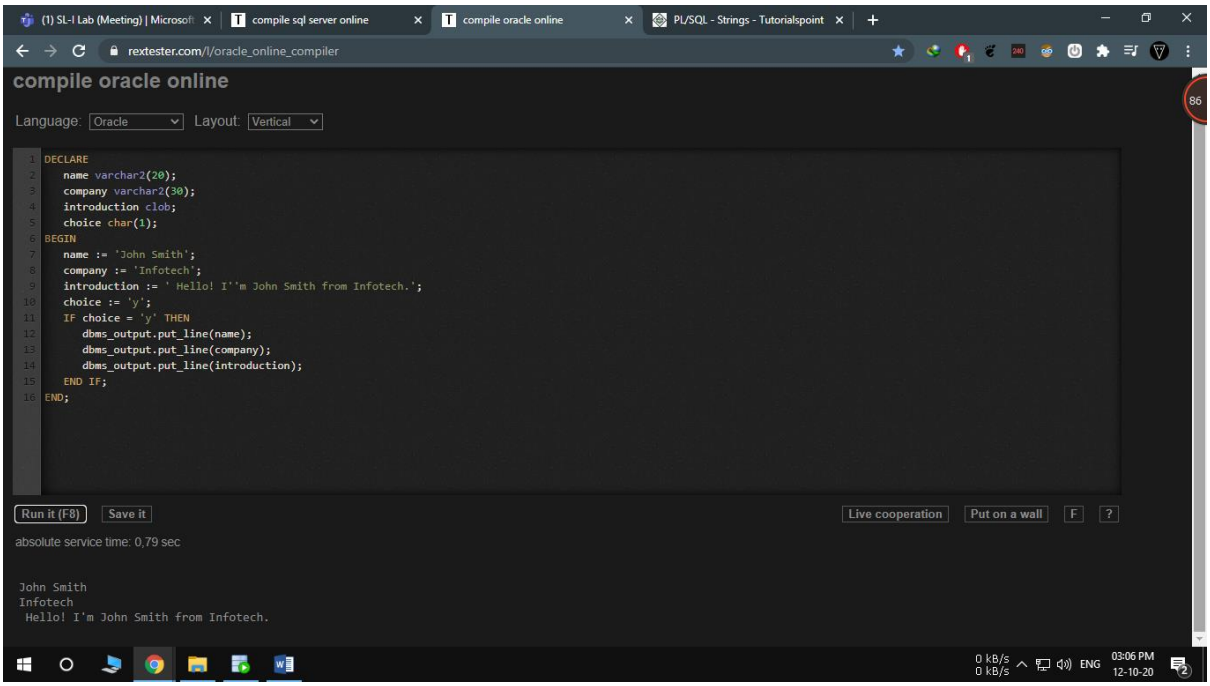


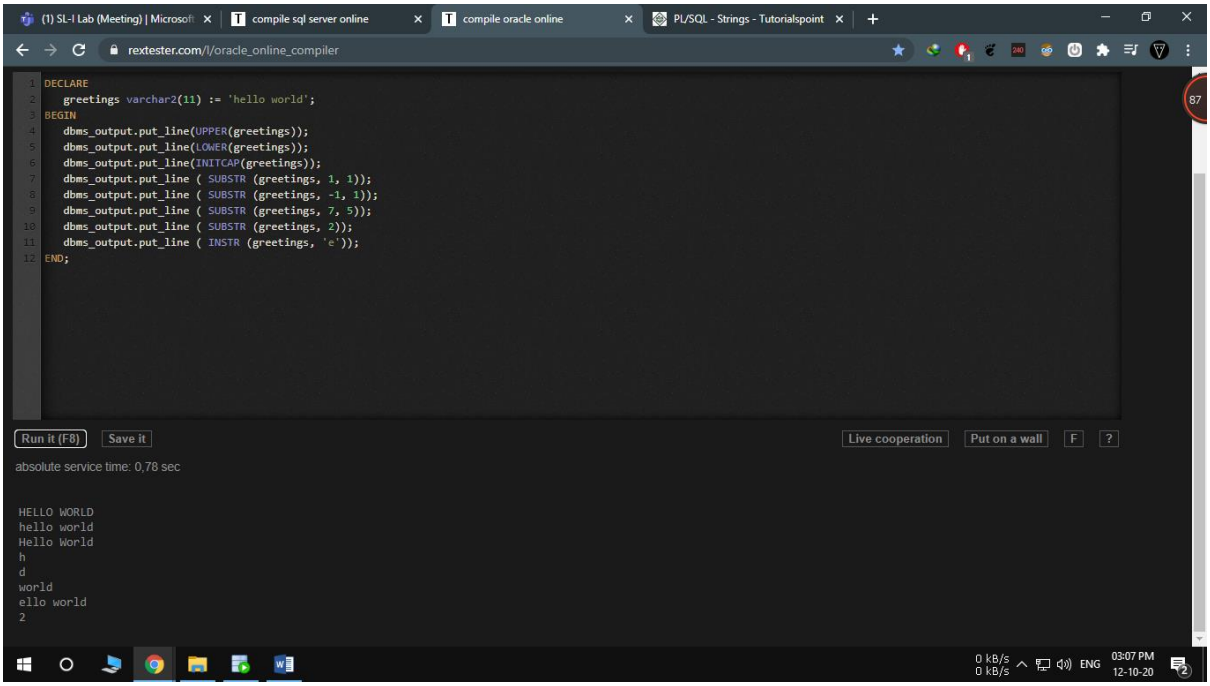



compile sql server online
compile sql server online
PL/SQL - Variables - Tutorialspoi
rextester.com/l/sql_server_online_compiler
SALARY DECIMAL (18, 2),
PRIMARY KEY (ID)
);
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00);
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (2, 'Khilan', 25, 'Delhi', 1500.00);
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (3, 'kaushik', 23, 'Kota', 2000.00);
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (4, 'Chaitali', 25, 'Mumbai', 6500.00);
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (5, 'Hardik', 27, 'Bhopal', 8500.00);
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (6, 'Komal', 22, 'MP', 4500.00);
select *from CUSTOMERS
Run it (F8)
Save it
View schema
Live cooperation
Put on a wall
F
?
Execution time: 0,02 sec, rows selected: 6, rows affected: 6, absolute service time: 0,19 sec, absolute service time: 0,19 sec
ID NAME AGE ADDRESS SALARY
1 1 Ramesh 32 Ahmedabad 2000,00
2 2 Khilan 25 Delhi 1500,00
3 3 kaushik 23 Kota 2000,00
4 4 Chaitali 25 Mumbai 6500,00
5 5 Hardik 27 Bhopal 8500,00
6 6 Komal 22 MP 4500,00
8 kB/s
149 kB/s
ENG
02:52 PM
12-10-20


(1) SL-I Lab (Meeting) | Microsoft
compile sql server online
compile oracle online
PL/SQL - Loops - Tutorialspoint
rextester.com/l/oracle_online_compiler
DECLARE
i number(1);
j number(1);
BEGIN
<< outer_loop >>
FOR i IN 1..3 LOOP
<< inner_loop >>
FOR j IN 1..3 LOOP
dbms_output.put_line('i is: '|| i || ' and j is: ' || j);
END loop inner_loop;
END loop outer_loop;
END;
Run it (F8)
Save it
Live cooperation
Put on a wall
F
?
absolute service time: 0,75 sec
i is: 1 and j is: 1
i is: 1 and j is: 2
i is: 1 and j is: 3
i is: 2 and j is: 1
i is: 2 and j is: 2
i is: 2 and j is: 3
i is: 3 and j is: 1
i is: 3 and j is: 2
i is: 3 and j is: 3
5 kB/s
14 kB/s
ENG
03:05 PM
12-10-20

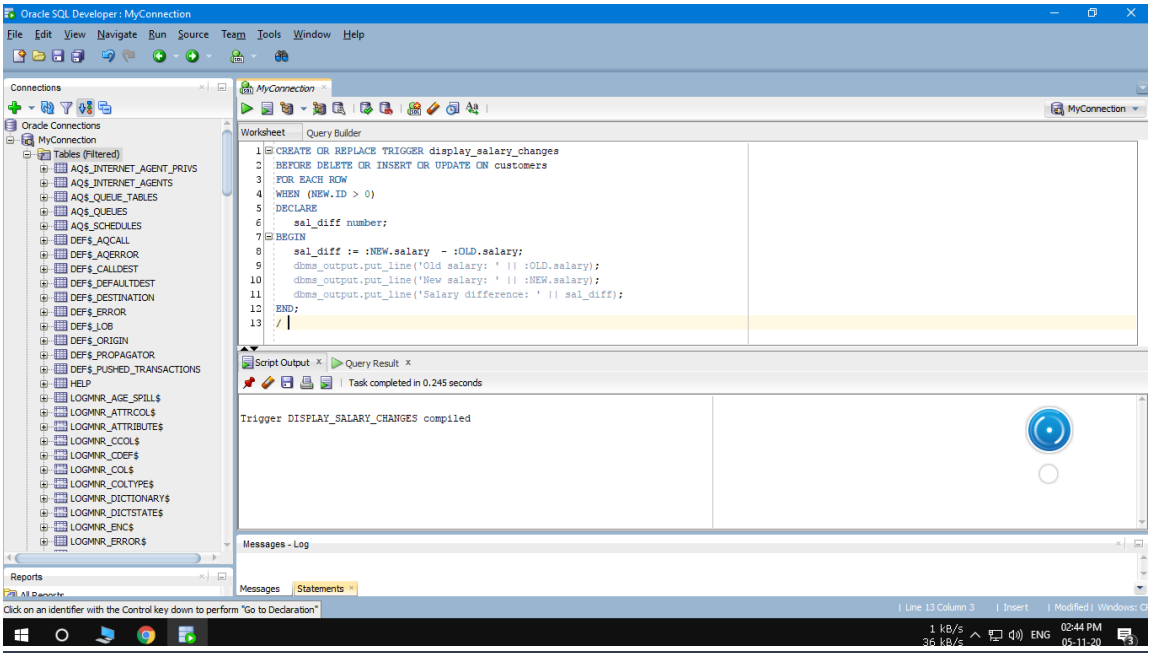

(1) SL-I Lab (Meeting) | Microsoft
compile sql server online
compile oracle online
PL/SQL - Loops - Tutorialspoint
rextester.com/l/oracle_online_compiler
DECLARE
i number(1);
j number(1);
BEGIN
<< outer_loop >>
FOR i IN 1..3 LOOP
<< inner_loop >>
FOR j IN 1..3 LOOP
dbms_output.put_line('i is: '|| i || ' and j is: ' || j);
END loop inner_loop;
END loop outer_loop;
END;
Run it (F8)
Save it
Live cooperation
Put on a wall
absolute service time: 0,75 sec
i is: 1 and j is: 1
i is: 1 and j is: 2
i is: 1 and j is: 3
i is: 2 and j is: 1
i is: 2 and j is: 2
i is: 2 and j is: 3
i is: 3 and j is: 1
i is: 3 and j is: 2
i is: 3 and j is: 3
ENG
03:05 PM
12-10-20

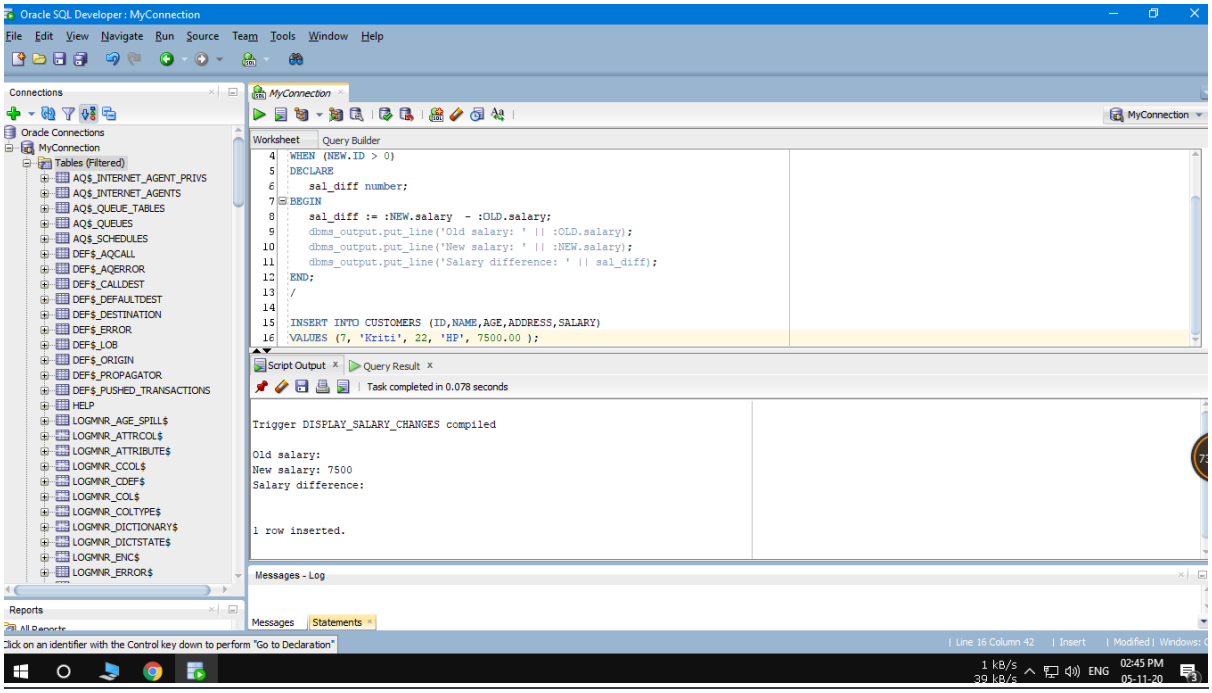

(1) SL-I Lab (Meeting) | Microsoft
compile sql server online
compile oracle online
PL/SQL - Strings - Tutorialspoint
rextester.com/l/oracle_online_compiler
compile oracle online
Language: Oracle
Layout: Vertical
DECLARE
name varchar2(20);
company varchar2(30);
introduction clob;
choice char(1);
BEGIN
name := 'John Smith';
company := 'Infotech';
introduction := ' Hello! I''m John Smith from Infotech.';
choice := 'y';
IF choice = 'y' THEN
dbms_output.put_line(name);
dbms_output.put_line(company);
dbms_output.put_line(introduction);
END IF;
END;
Run it (F8)
Save it
Live cooperation
Put on a wall
absolute service time: 0,79 sec
John Smith
Infotech
Hello! I'm John Smith from Infotech.
ENG
03:06 PM
12-10-20

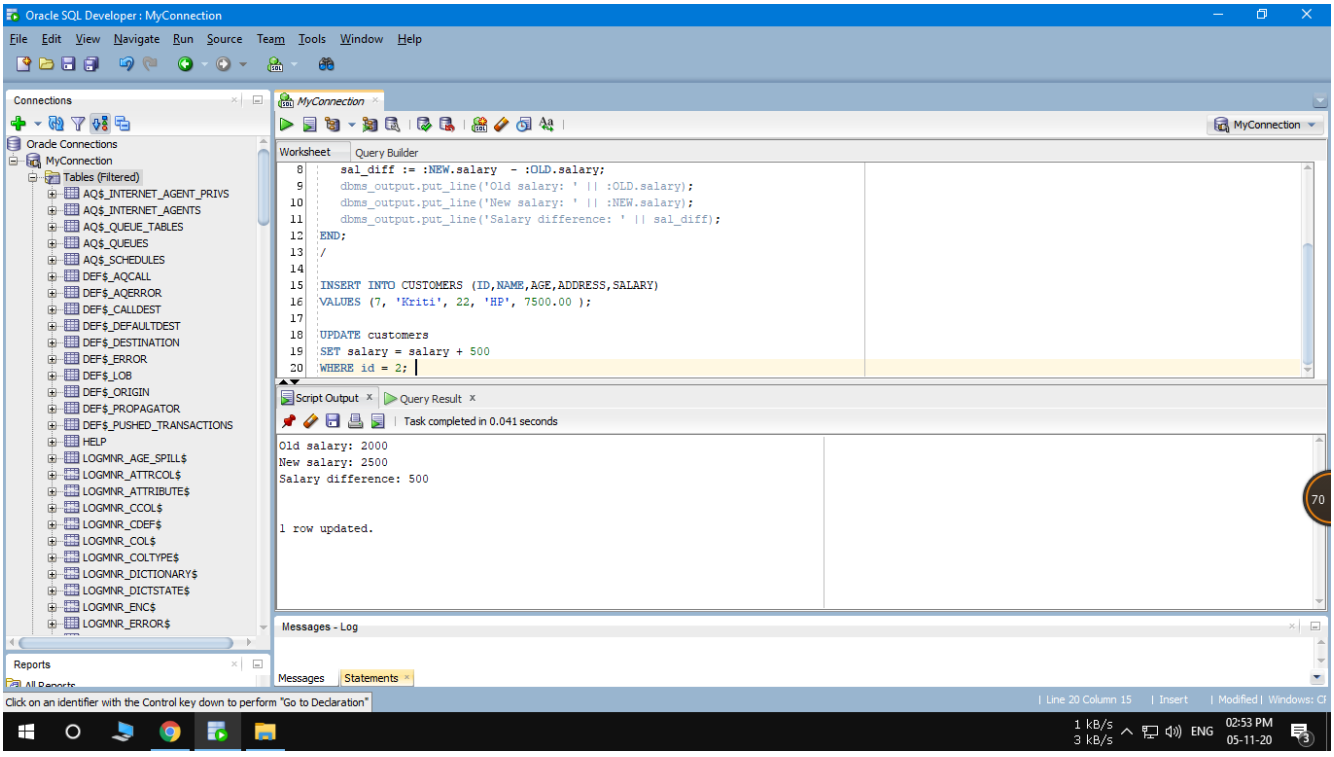

(1) SL-I Lab (Meeting) | Microsoft
compile sql server online
compile oracle online
PL/SQL - Strings - Tutorialspoint
rextester.com/l/oracle_online_compiler
DECLARE
greetings varchar2(11) := 'hello world';
BEGIN
dbms_output.put_line(UPPER(greetings));
dbms_output.put_line(LOWER(greetings));
dbms_output.put_line(INITCAP(greetings));
dbms_output.put_line (SUBSTR (greetings, 1, 1));
dbms_output.put_line (SUBSTR (greetings, -1, 1));
dbms_output.put_line (SUBSTR (greetings, 7, 5));
dbms_output.put_line (SUBSTR (greetings, 2));
dbms_output.put_line (INSTR (greetings, 'e'));
END;
Run it (F8)
Save it
Live cooperation
Put on a wall
F
?
absolute service time: 0,78 sec
HELLO WORLD
hello world
Hello World
h
d
world
ello world
2
ENG
03:07 PM
12-10-20

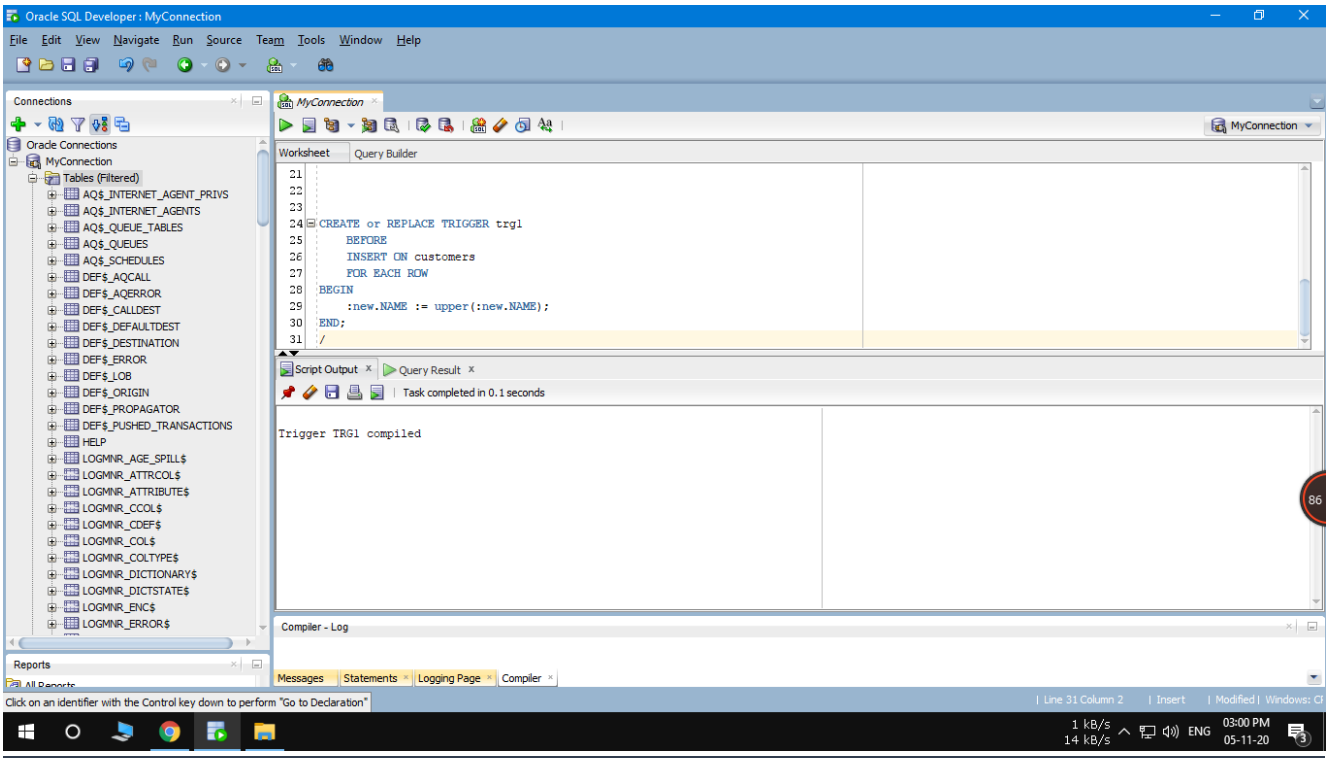
Group B Assignment 5 5/11/2020

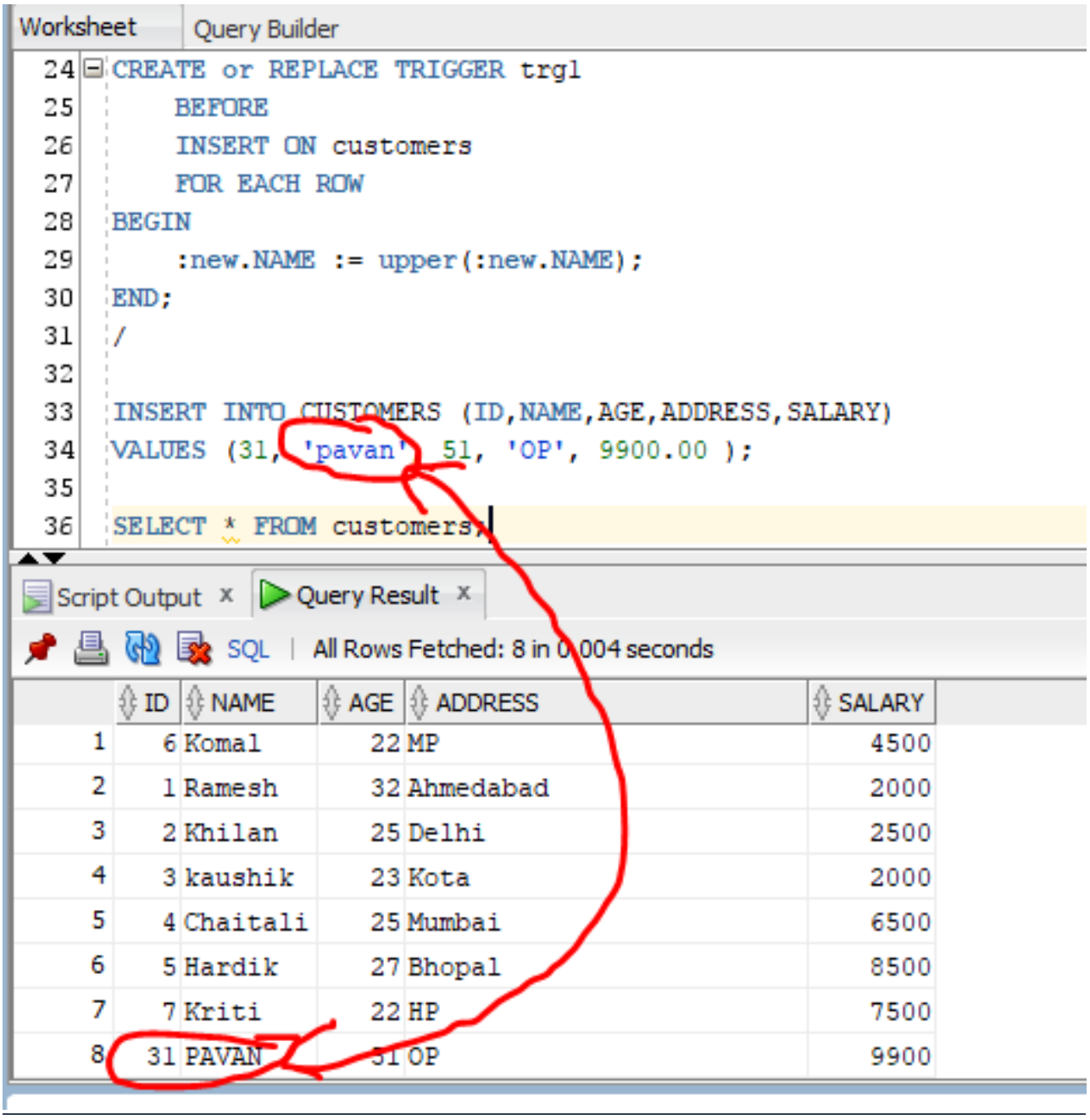
PAVAN S. PATIL
ROLLNO. 51

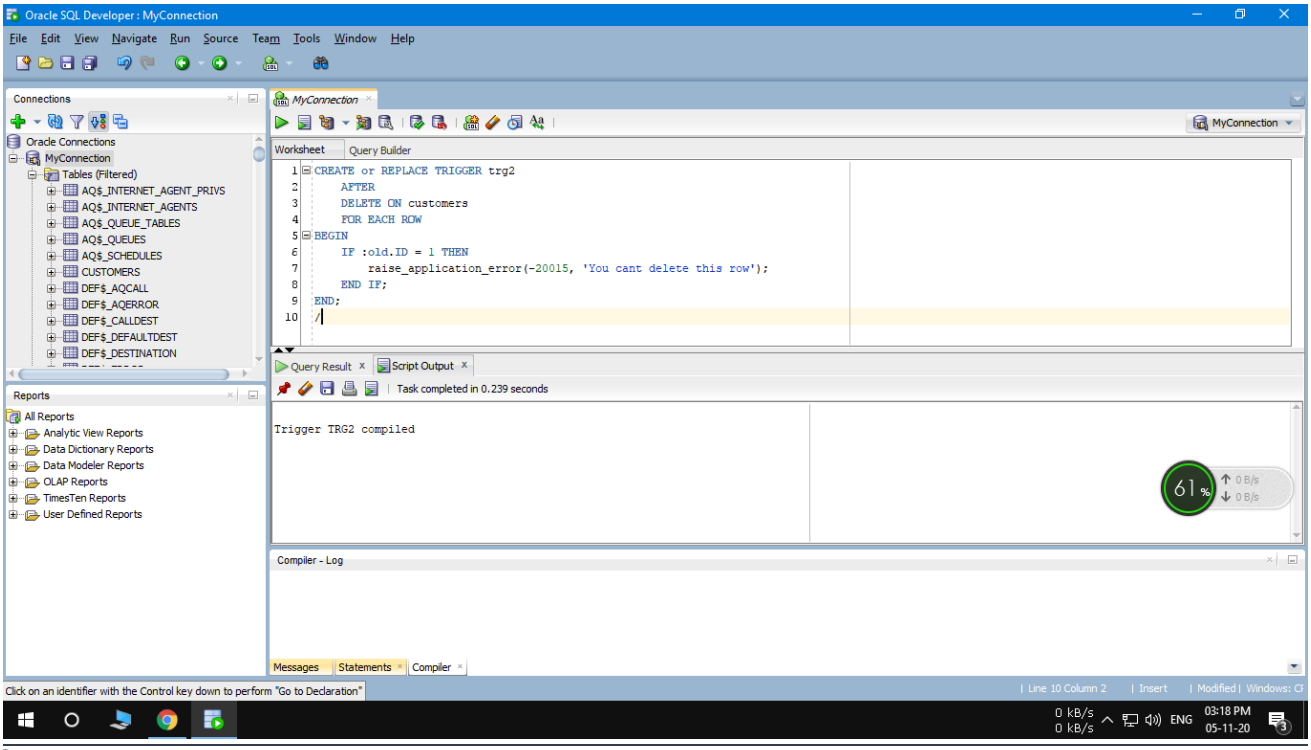





Oracle SQL Developer : MyConnection
File Edit View Navigate Run Source Team Tools Window Help
Connections
Oracle Connections
MyConnection
Tables (Filtered)
AQ$_INTERNET_AGENT_PRIVS
AQ$_INTERNET_AGENTS
AQ$_QUEUE_TABLES
AQ$_QUEUES
AQ$_SCHEDULES
DEF$_AQCALL
DEF$_AQERROR
DEF$_CALLDEST
DEF$_DEFAULTDEST
DEF$_DESTINATION
DEF$_ERROR
DEF$_LOB
DEF$_ORIGIN
DEF$_PROPAGATOR
DEF$_PUSHED_TRANSACTIONS
HELP
LOGMNR_AGE_SPILL$
LOGMNR_ATTRCOL$
LOGMNR_ATTRIBUTE$
LOGMNR_CCOL$
LOGMNR_CDEF$
LOGMNR_COL$
LOGMNR_COLTYPE$
LOGMNR_DICTIONARY$
LOGMNR_DICTSTATE$
LOGMNR_ENC$
LOGMNR_ERROR$
Reports
MyConnection
Worksheet
Query Builder
8 sal_diff := :NEW.salary - :OLD.salary;
9 dbms_output.put_line('Old salary: ' || :OLD.salary);
10 dbms_output.put_line('New salary: ' || :NEW.salary);
11 dbms_output.put_line('Salary difference: ' || sal_diff);
12 END;
13 /
14
15 INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
16 VALUES (7, 'Kriti', 22, 'HP', 7500.00);
17
18 UPDATE customers
19 SET salary = salary + 500
20 WHERE id = 2;
Script Output
Query Result
Task completed in 0.041 seconds
Old salary: 2000
New salary: 2500
Salary difference: 500
1 row updated.
Messages - Log
Messages
Statements
Click on an identifier with the Control key down to perform "Go to Declaration"
Line 20 Column 15
Insert
Modified
Windows: Cr
1 kB/s
3 kB/s
ENG
02:53 PM
05-11-20


Oracle SQL Developer : MyConnection
File Edit View Navigate Run Source Team Tools Window Help
Connections
Oracle Connections
MyConnection
Tables (Filtered)
AQ$_INTERNET_AGENT_PRIVS
AQ$_INTERNET_AGENTS
AQ$_QUEUE_TABLES
AQ$_QUEUES
AQ$_SCHEDULES
DEF$_AQCALL
DEF$_AQERROR
DEF$_CALLDEST
DEF$_DEFAULTDEST
DEF$_DESTINATION
DEF$_ERROR
DEF$_LOB
DEF$_ORIGIN
DEF$_PROPAGATOR
DEF$_PUSHED_TRANSACTIONS
HELP
LOGMNR_AGE_SPILL$
LOGMNR_ATTRCOL$
LOGMNR_ATTRIBUTE$
LOGMNR_CCOL$
LOGMNR_CDEF$
LOGMNR_COL$
LOGMNR_COLTYPE$
LOGMNR_DICTIONARY$
LOGMNR_DICTSTATE$
LOGMNR_ENC$
LOGMNR_ERROR$
Reports
MyConnection
Worksheet
Query Builder
21
22
23
24 CREATE or REPLACE TRIGGER trg1
25 BEFORE
26 INSERT ON customers
27 FOR EACH ROW
28 BEGIN
29 :new.NAME := upper(:new.NAME);
30 END;
31 /
Script Output
Query Result
Task completed in 0.1 seconds
Trigger TRG1 compiled
Compiler - Log
Messages
Statements
Logging Page
Compiler
Click on an identifier with the Control key down to perform "Go to Declaration"
Line 31 Column 2
Insert
Modified
Windows: Cr
1 kB/s
14 kB/s
ENG
03:00 PM
05-11-20


Worksheet
Query Builder
CREATE or REPLACE TRIGGER trg1
BEFORE
INSERT ON customers
FOR EACH ROW
BEGIN
:new.NAME := upper(:new.NAME);
END;
/
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (31, 'pavan', 51, 'OP', 9900.00);
SELECT * FROM customers;
Script Output
Query Result
SQL | All Rows Fetched: 8 in 0.004 seconds
ID NAME AGE ADDRESS SALARY
1 6 Komal 22 MP 4500
2 1 Ramesh 32 Ahmedabad 2000
3 2 Khilan 25 Delhi 2500
4 3 kaushik 23 Kota 2000
5 4 Chaitali 25 Mumbai 6500
6 5 Hardik 27 Bhopal 8500
7 7 Kriti 22 HP 7500
8 31 PAVAN 51 OP 9900



Worksheet Query Builder

```
1  CREATE or REPLACE TRIGGER trg2
2      AFTER
3      DELETE ON customers
4      FOR EACH ROW
5  BEGIN
6      IF :old.ID = 1 THEN
7          raise_application_error(-20015, 'You cant delete this row');
8      END IF;
9  END;
10 /
11 delete from customers where ID = 1;
```

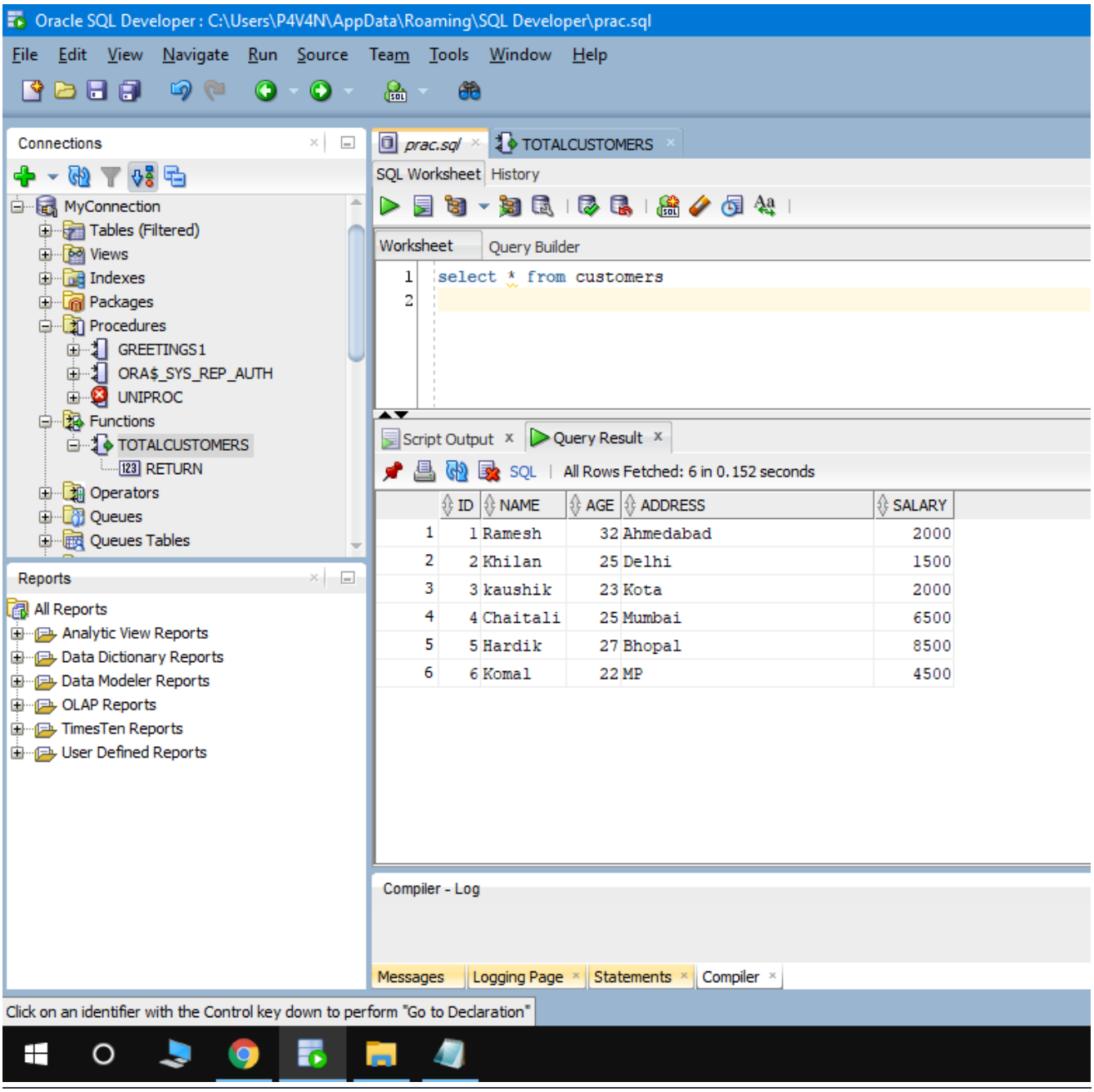
Query Result | Script Output

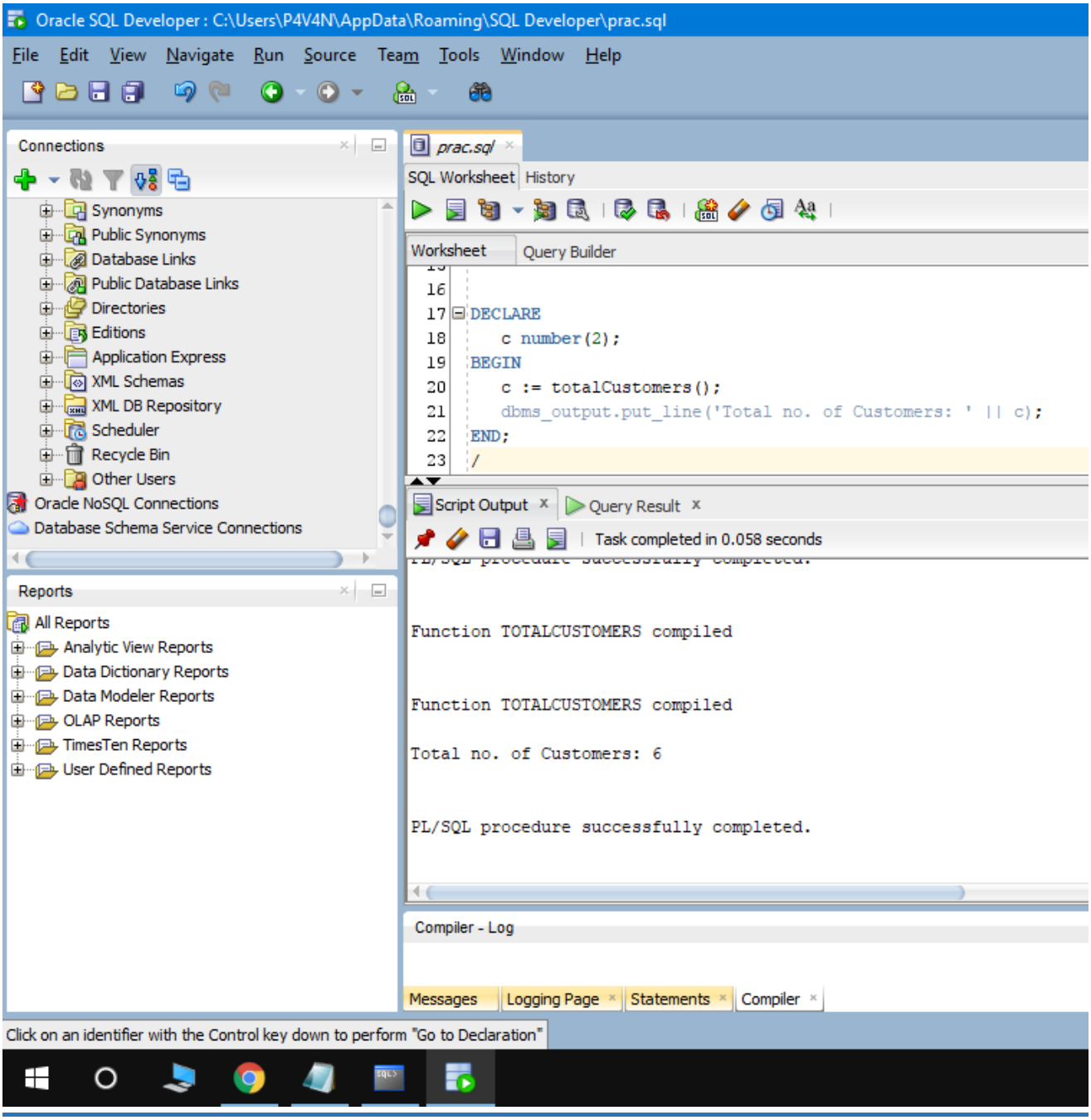
Task completed in 0.245 seconds

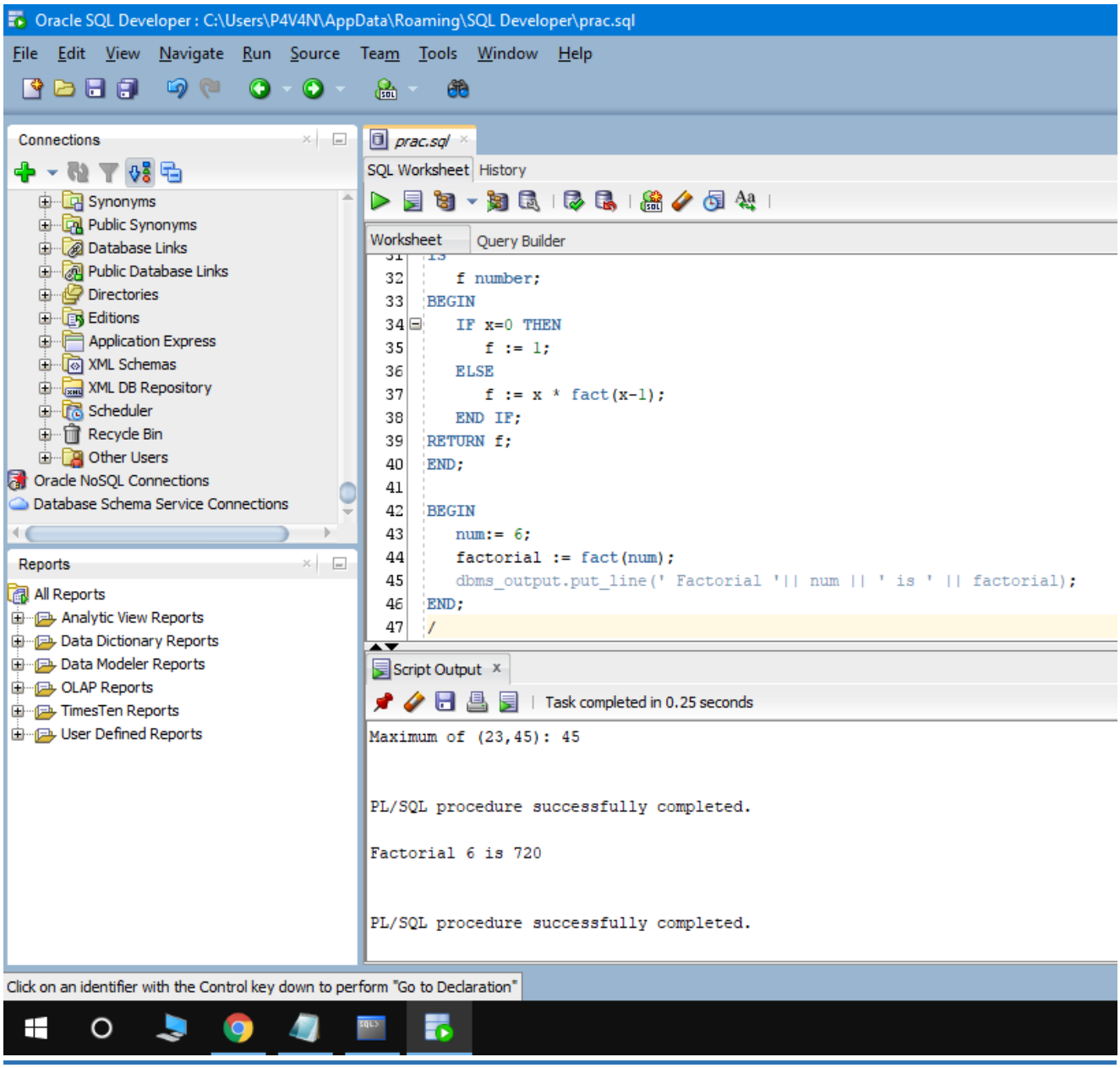
```
Error starting at line : 11 in command -
delete from customers where ID = 1
Error report -
ORA-20015: You cant delete this row
ORA-06512: at "SYSTEM.TRG2", line 3
ORA-04088: error during execution of trigger 'SYSTEM.TRG2'
```

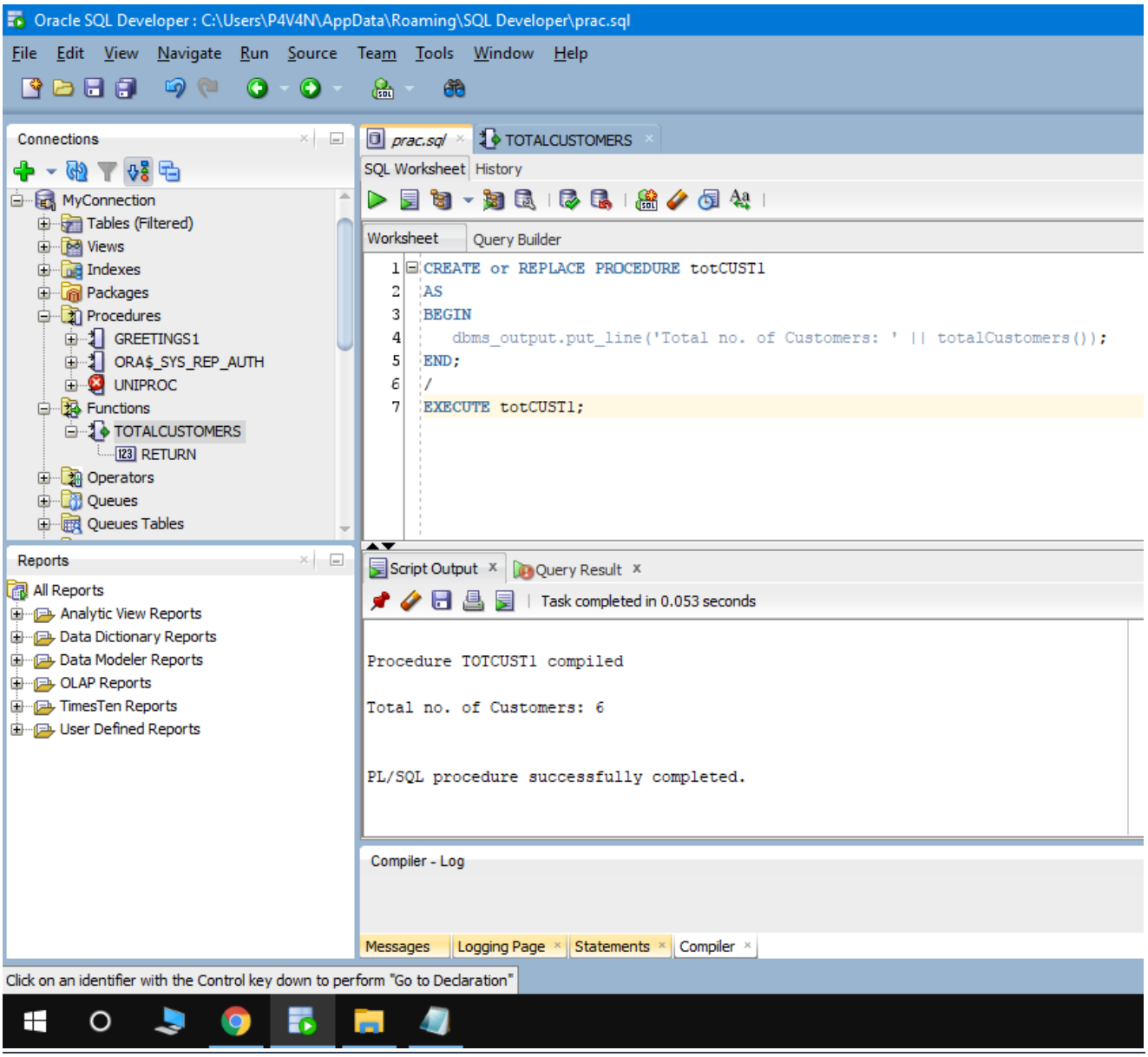
Group B Assignment 6 22/10/2020

PAVAN S. PATIL
ROLLNO. 51




Oracle SQL Developer : C:\Users\P4V4N\AppData\Roaming\SQL Developer\prac.sql
File Edit View Navigate Run Source Team Tools Window Help
Connections
Synonyms
Public Synonyms
Database Links
Public Database Links
Directories
Editions
Application Express
XML Schemas
XML DB Repository
Scheduler
Recycle Bin
Other Users
Oracle NoSQL Connections
Database Schema Service Connections
Reports
All Reports
Analytic View Reports
Data Dictionary Reports
Data Modeler Reports
OLAP Reports
TimesTen Reports
User Defined Reports
prac.sql
SQL Worksheet History
Worksheet Query Builder
16
17 DECLARE
18 c number(2);
19 BEGIN
20 c := totalCustomers();
21 dbms_output.put_line('Total no. of Customers: ' || c);
22 END;
23 /
Script Output
Query Result
Task completed in 0.058 seconds
Function TOTALCUSTOMERS compiled
Function TOTALCUSTOMERS compiled
Total no. of Customers: 6
PL/SQL procedure successfully completed.
Compiler - Log
Messages
Logging Page
Statements
Compiler
Click on an identifier with the Control key down to perform "Go to Declaration"


Oracle SQL Developer : C:\Users\P4V4N\AppData\Roaming\SQL Developer\prac.sql
File Edit View Navigate Run Source Team Tools Window Help
Connections
Synonyms
Public Synonyms
Database Links
Public Database Links
Directories
Editions
Application Express
XML Schemas
XML DB Repository
Scheduler
Recycle Bin
Other Users
Oracle NoSQL Connections
Database Schema Service Connections
Reports
All Reports
Analytic View Reports
Data Dictionary Reports
Data Modeler Reports
OLAP Reports
TimesTen Reports
User Defined Reports
prac.sql
SQL Worksheet History
Worksheet Query Builder
32 f number;
33 BEGIN
34 IF x=0 THEN
35 f := 1;
36 ELSE
37 f := x * fact(x-1);
38 END IF;
39 RETURN f;
40 END;
41
42 BEGIN
43 num:= 6;
44 factorial := fact(num);
45 dbms_output.put_line(' Factorial '|| num || ' is ' || factorial);
46 END;
47 /
Script Output
Task completed in 0.25 seconds
Maximum of (23,45): 45
PL/SQL procedure successfully completed.
Factorial 6 is 720
PL/SQL procedure successfully completed.
Click on an identifier with the Control key down to perform "Go to Declaration"


Oracle SQL Developer : C:\Users\P4V4N\AppData\Roaming\SQL Developer\prac.sql
File Edit View Navigate Run Source Team Tools Window Help
Connections
MyConnection
Tables (Filtered)
Views
Indexes
Packages
Procedures
GREETINGS1
ORA$_SYS_REP_AUTH
UNIPROC
Functions
TOTALCUSTOMERS
RETURN
Operators
Queues
Queues Tables
Reports
All Reports
Analytic View Reports
Data Dictionary Reports
Data Modeler Reports
OLAP Reports
TimesTen Reports
User Defined Reports
prac.sql
TOTALCUSTOMERS
SQL Worksheet History
Worksheet Query Builder
1 CREATE or REPLACE PROCEDURE totCUST1
2 AS
3 BEGIN
4 dbms_output.put_line('Total no. of Customers: ' || totalCustomers());
5 END;
6 /
7 EXECUTE totCUST1;
Script Output
Query Result
Task completed in 0.053 seconds
Procedure TOTCUST1 compiled
Total no. of Customers: 6
PL/SQL procedure successfully completed.
Compiler - Log
Messages Logging Page Statements Compiler
Click on an identifier with the Control key down to perform "Go to Declaration"

GROUP C |Assignment - 1

Name : Pavan Patil
Rollno. : 51

1. Create a database with suitable example using MongoDB and implement
   - Inserting and saving document (batch insert, insert validation)
   - Removing document
   - Updating document (document replacement, using modifiers, upserts, updating documents, returning updated documents)

Create Database

```
> use emp_db switched to db emp_db

> db.createCollection("employees")
{ "ok" : 1 }
> show collections employees
```

Insert Document

```
> db.employees.insert(
... {
... "ID" : "1",
... "Name" : "Pavan",
... "Designation" : "CEO"
... })
WriteResult({ "nInserted" : 1 })
> db.employees.find()
{ "_id" : ObjectId("5d8e4ab54f5e86a76906931a"), "ID" : "1",
"Name" : "Pavan", "Designation" : "CEO" }
```

Batch Insert

```
>db.employees.ins
ert( [ { "ID" : "3",
"Name" : "Karan",
"Designation" : "Product Manager" }, {"ID" : "4", "Name" :
"Rohit", "Designation" : "Vice President"} ] )
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

Remove Document

```
> db.employees.remove({ID:"5"})
WriteResult({ "nRemoved" : 1 })

> db.employees.find({})
{ "_id" : ObjectId("5d8e4ab54f5e86a76906931a"), "ID" : "1",
"Name" : "Pavan", "Designation" : "CEO" }
{ "_id" : ObjectId("5d8e4cf44f5e86a76906931b"), "ID" : "2",
"Name" : "Gaurav", "Designation" : "CFO" }
{ "_id" : ObjectId("5d8e4d844f5e86a76906931c"), "ID" : "3",
"Name" : "Karan", "Designation" : "Product Manager" }
{ "_id" : ObjectId("5d8e4d844f5e86a76906931d"), "ID" : "4",
"Name" : "Rohit", "Designation" : "Vice President" }
```

Update Document

```
> db.employees.update({Designation : "Vice President"}, {$set: {Designation: "Senior VP"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1
})
> db.employees.find({})
{ "_id" : ObjectId("5d8e4ab54f5e86a76906931a"), "ID" : "1",
"Name" : "Pavan", "Designation" : "CEO" }
{ "_id" : ObjectId("5d8e4cf44f5e86a76906931b"), "ID" : "2",
"Name" : "Gaurav", "Designation" : "CFO" }
{ "_id" : ObjectId("5d8e4d844f5e86a76906931c"), "ID" : "3",
"Name" : "Karan", "Designation" : "Product Manager" }
{ "_id" : ObjectId("5d8e4d844f5e86a76906931d"), "ID" : "4",
"Name" : "Rohit", "Designation" : "Senior VP" }
{ "_id" : ObjectId("5d8edba14f5e86a76906931f"), "ID" : "5",
"Name" : "Rushikesh", "Designation" : "Product Designer" }
{ "_id" : ObjectId("5d8edbba4f5e86a769069320"), "ID" : "6",
"Name" : "Gopal", "Designation" : "COO" }
```

Insert field

```
> db.employees.update({Designation : "Software Engineer"},
{$set: {Skills: ["Python", "DBMS", "Java"]}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1
})
```

Delete field

```
> db.employees.update({Designation : "Software Engineer"},
{$unset: {Skills: []}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1
```

})

Update multiple documents

```
> db.employees.update({Designation : "Software Engineer"},
{$set: {Skills: ["Python", "DBMS", "Java"]}}, {multi : true})
WriteResult({ "nMatched" : 2, "nUpserted" : 0, "nModified" : 2
})
> db.employees.find({})
{ "_id" : ObjectId("5d8e4ab54f5e86a76906931a"), "ID" : "1",
"Name" : "Pavan", "Designation" : "CEO" }
{ "_id" : ObjectId("5d8e4cf44f5e86a76906931b"), "ID" : "2",
"Name" : "Gaurav", "Designation" : "CFO" }
{ "_id" : ObjectId("5d8e4d844f5e86a76906931c"), "ID" : "3",
"Name" : "Karan", "Designation" : "Product Manager" }
{ "_id" : ObjectId("5d8e4d844f5e86a76906931d"), "ID" : "4",
"Name" : "Rohit", "Designation" : "Senior VP" }
{ "_id" : ObjectId("5d8edba14f5e86a76906931f"), "ID" : "5",
"Name" : "Rushikesh", "Designation" : "Product Designer" }
{ "_id" : ObjectId("5d8edbba4f5e86a769069320"), "ID" : "6",
"Name" : "Gopal", "Designation" : "COO" }
{ "_id" : ObjectId("5d8ede384f5e86a769069321"), "ID" : "7",
"Name" : "Mohit", "Designation" : "Software Engineer", "Skills" : [ "Python", "DBMS", "Java" ] }
{ "_id" : ObjectId("5d8ede554f5e86a769069322"), "ID" : "8",
"Name" : "Kunal", "Designation" : "Software Engineer", "Skills"
: [ "Python", "DBMS", "Java" ] }
```

Upsert

```
> db.employees.update({Designation: "Strategist"}, {Designation:
"CMO"}, {upsert: true}) WriteResult({
        "nMatched" : 0,
        "nUpserted" : 1, "nModified" : 0,
        "_id" : ObjectId("5d8ee28626dd63ef91974468")
})
```

```
> db.employees.find({}).pretty()
{
        "_id" : ObjectId("5d8e4ab54f5e86a76906931a"),
        "ID" : "1",
        "Name" : "Pavan",
        "Designation" : "CEO"
} {
        "_id" : ObjectId("5d8e4cf44f5e86a76906931b"),
        "ID" : "2",
        "Name" : "Gaurav",
        "Designation" : "CFO"
} {
        "_id" : ObjectId("5d8e4d844f5e86a76906931c"),
        "ID" : "3",
        "Name" : "Karan",
        "Designation" : "Product Manager"
} {
        "_id" : ObjectId("5d8e4d844f5e86a76906931d"),
        "ID" : "4",
        "Name" : "Rohit",
        "Designation" : "Senior VP"
} {
        "_id" : ObjectId("5d8edba14f5e86a76906931f"),
        "ID" : "5",
        "Name" : "Rushikesh",
        "Designation" : "Product Designer"
} {
        "_id" : ObjectId("5d8edbba4f5e86a769069320"),
        "ID" : "6",
        "Name" : "Gopal",
        "Designation" : "COO"
} {
        "_id" : ObjectId("5d8ede384f5e86a769069321"),
        "ID" : "7",
        "Name" : "Mohit",
        "Designation" : "Software Engineer", "Skills" : [
                "Python",
                "DBMS",
```

```
                "Java"
        ] } {
        "_id" : ObjectId("5d8ede554f5e86a769069322"),
        "ID" : "8",
        "Name" : "Kunal",
        "Designation" : "Software Engineer", "Skills" : [
                "Python",
                "DBMS",
                "Java"
        ] }
{ "_id" : ObjectId("5d8ee28626dd63ef91974468"), "Designation" :
"CMO" }
```

GROUP C |Assignment - 2

Name : Pavan Patil
Rollno. : 51

2. Execute at least 10 queries on any suitable MongoDB database
that demonstrates following querying techniques:

- find and findOne (specific values)
- Query criteria (Query conditionals, OR queries, $not, Conditional semantics)
- Type-specific queries (Null, Regular expression, Querying arrays)

Find document

Display the list of all employees

```
> db.employees.find({})
{ "_id" : ObjectId("5d8e4ab54f5e86a76906931a"), "ID" : "1",
"Name" : "Pavan", "Designation" : "CEO" }
{ "_id" : ObjectId("5d8e4cf44f5e86a76906931b"), "ID" : "2",
"Name" : "Karan", "Designation" : "CFO" }
{ "_id" : ObjectId("5d8e4d844f5e86a76906931c"), "ID" : "3",
"Name" : "Hrushikesh", "Designation" : "Product Manager" }
{ "_id" : ObjectId("5d8e4d844f5e86a76906931d"), "ID" : "4",
"Name" : "Rohit", "Designation" : "Senior VP" }
{ "_id" : ObjectId("5d8edba14f5e86a76906931f"), "ID" : "5",
"Name" : "Gaurav", "Designation" : "Product Designer" }
{ "_id" : ObjectId("5d8edbba4f5e86a769069320"), "ID" : "6",
"Name" : "Gopal", "Designation" : "COO" }
{ "_id" : ObjectId("5d8ede384f5e86a769069321"), "ID" : "7",
"Name" : "Jay", "Designation" : "Software Engineer", "Skills" : [ "Python", "DBMS", "Java" ] }
{ "_id" : ObjectId("5d8ede554f5e86a769069322"), "ID" : "8",
"Name" : "Arnab", "Designation" : "Software Engineer", "Skills"
: [ "Python", "DBMS", "Java" ] }
Display employee details having ID=1
> db.employees.find({ID: "1"})
{ "_id" : ObjectId("5d8e4ab54f5e86a76906931a"), "ID" : "1",
```

"Name" : "Pavan", "Designation" : "CEO" }

Display employees who are software engineers

```
> db.employees.find({Designation: "Software Engineer"})
{ "_id" : ObjectId("5d8ede384f5e86a769069321"), "ID" : "7",
"Name" : "Jay", "Designation" : "Software Engineer", "Skills" : [ "Python", "DBMS", "Java" ] }
{ "_id" : ObjectId("5d8ede554f5e86a769069322"), "ID" : "8",
"Name" : "Arnab", "Designation" : "Software Engineer", "Skills"
: [ "Python", "DBMS", "Java" ] }
```

Find document in JSON format

```
> db.employees.find({}).pretty()
{
        "_id" : ObjectId("5d8e4ab54f5e86a76906931a"),
        "ID" : "1",
        "Name" : "Pavan",
        "Designation" : "CEO"
} {
        "_id" : ObjectId("5d8e4cf44f5e86a76906931b"),
        "ID" : "2",
        "Name" : "Karan",
        "Designation" : "CFO"
} {
        "_id" : ObjectId("5d8e4d844f5e86a76906931c"),
        "ID" : "3",
        "Name" : "Hrushikesh",
        "Designation" : "Product Manager"
} {
        "_id" : ObjectId("5d8e4d844f5e86a76906931d"),
        "ID" : "4",
        "Name" : "Rohit",
        "Designation" : "Senior VP"
} {
        "_id" : ObjectId("5d8edba14f5e86a76906931f"),
        "ID" : "5",
        "Name" : "Gaurav",
        "Designation" : "Product Designer" }
{
```

```
        "_id" : ObjectId("5d8edbba4f5e86a769069320"),
        "ID" : "6",
        "Name" : "Gopal",
        "Designation" : "COO"
} {
        "_id" : ObjectId("5d8ede384f5e86a769069321"),
        "ID" : "7",
        "Name" : "Jay",
        "Designation" : "Software Engineer", "Skills" : [
                "Python",
                "DBMS",
                "Java"
        ] }
{
        "_id" : ObjectId("5d8ede554f5e86a769069322"),
        "ID" : "8",
        "Name" : "Arnab",
        "Designation" : "Software Engineer", "Skills" : [
                "Python",
                "DBMS",
                "Java"
        ]
}
```


Use of findOne

Display the first document of employee working as software engineer


```
> db.employees.findOne({Designation: "Software Engineer"})
{
        "_id" : ObjectId("5d8ede384f5e86a769069321"),
        "ID" : "7",
        "Name" : "Jay",
        "Designation" : "Software Engineer", "Skills" : [
                "Python",
                "DBMS",
                "Java" ]
```

}

AND condition

Display employees working as a software engineer and having Java skills

```
> db.employees.find({Designation: "Software Engineer", Skills: "Java"})
{ "_id" : ObjectId("5d8ede384f5e86a769069321"), "ID" : "7",
"Name" : "Jay", "Designation" : "Software Engineer", "Skills" : [ "Python", "DBMS", "Java" ] }
{ "_id" : ObjectId("5d8ede554f5e86a769069322"), "ID" : "8",
"Name" : "Arnab", "Designation" : "Software Engineer", "Skills"
: [ "Python", "DBMS", "Java" ] }
{ "_id" : ObjectId("5d8f06934f5e86a769069325"), "ID" : "11",
"Name" : "Andy", "Designation" : "Software Engineer", "Skills" :
[ "Java", "C++", ".NET" ] }
```

Display employees working as product manager and having Agile
skills

```
> db.employees.find({Designation: "Product Manager", Skills:
"Agile"})
>
```

OR condition

Display employees working as a software engineer or a product manager

```
> db.employees.find({$or: [{Designation: "Software
Engineer"},{Designation: "Product Manager"}]})
{ "_id" : ObjectId("5d8e4d844f5e86a76906931c"), "ID" : "3",
"Name" : "Hrushikesh", "Designation" : "Product Manager" }
{ "_id" : ObjectId("5d8ede384f5e86a769069321"), "ID" : "7",
"Name" : "Jay", "Designation" : "Software Engineer", "Skills" : [ "Python", "DBMS", "Java" ] }
{ "_id" : ObjectId("5d8ede554f5e86a769069322"), "ID" : "8",
"Name" : "Arnab", "Designation" : "Software Engineer", "Skills"
```

: ["Python", "DBMS", "Java"] }
Display employees working as a software engineer or having Java
skills

> db.employees.find({$or: [{Designation: "Software Engineer"},
{Skills: "Java"}]})
{ "_id" : ObjectId("5d8e4d844f5e86a76906931c"), "ID" : "3",
"Name" : "Hrushikesh", "Designation" : "Product Manager", "Skills" :
["Management", "SCRUM", "Java"] }
{ "_id" : ObjectId("5d8ede384f5e86a769069321"), "ID" : "7",
"Name" : "Jay", "Designation" : "Software Engineer", "Skills" : ["Python", "DBMS", "Java"] }
{ "_id" : ObjectId("5d8ede554f5e86a769069322"), "ID" : "8",
"Name" : "Arnab", "Designation" : "Software Engineer", "Skills"
: ["Python", "DBMS", "Java"] }
{ "_id" : ObjectId("5d8f06934f5e86a769069324"), "ID" : "10",
"Name" : "Pam", "Designation" : "Web Developer", "Skills" : [
"HTML", "Bootstrap", "PHP", "Javascript", "React", "Java"] }
{ "_id" : ObjectId("5d8f06934f5e86a769069325"), "ID" : "11",
"Name" : "Andy", "Designation" : "Software Engineer", "Skills" :
["Java", "C++", ".NET"] }

NOT condition

Display employees not having DBMS skills

> db.employees.find({Skills: {$not: {$eq: "DBMS"}}})
{ "_id" : ObjectId("5d8e4ab54f5e86a76906931a"), "ID" : "1",
"Name" : "Pavan", "Designation" : "CEO", "Skills" : ["Management", "Strategy"] }
{ "_id" : ObjectId("5d8e4cf44f5e86a76906931b"), "ID" : "2",
"Name" : "Karan", "Designation" : "CFO", "Skills" : [
"Financial Analysis", "Accounting"] }
{ "_id" : ObjectId("5d8e4d844f5e86a76906931c"), "ID" : "3",
"Name" : "Hrushikesh", "Designation" : "Product Manager", "Skills" :
["Management", "SCRUM", "Java"] }
{ "_id" : ObjectId("5d8e4d844f5e86a76906931d"), "ID" : "4", "Name" : "Rohit", "Designation" :
"Senior VP", "Skills" : [

"PHP", ".NET"] }
{ "_id" : ObjectId("5d8edba14f5e86a76906931f"), "ID" : "5",
"Name" : "Gaurav", "Designation" : "Product Designer", "Skills"
: ["Prototyping", "CAD"] }
{ "_id" : ObjectId("5d8edbba4f5e86a769069320"), "ID" : "6",
"Name" : "Gopal", "Designation" : "COO" }
{ "_id" : ObjectId("5d8f06934f5e86a769069323"), "ID" : "9",
"Name" : "Mohit", "Designation" : "CMO", "Skills" : [
"Strategy", "Copywriting"] }
{ "_id" : ObjectId("5d8f06934f5e86a769069324"), "ID" : "10",
"Name" : "Pam", "Designation" : "Web Developer", "Skills" : [
"HTML", "Bootstrap", "PHP", "Javascript", "React", "Java"] }
{ "_id" : ObjectId("5d8f06934f5e86a769069325"), "ID" : "11",
"Name" : "Andy", "Designation" : "Software Engineer", "Skills" :
["Java", "C++", ".NET"] }

Relational operators

List all employees having salary greater than 150000

> db.employees.find({Salary: {$gt: 150000}})
{ "_id" : ObjectId("5d8e4ab54f5e86a76906931a"), "ID" : "1",
"Name" : "Pavan", "Designation" : "CEO", "Skills" : [
"Management", "Strategy"], "Salary" : 250000 }
{ "_id" : ObjectId("5d8e4cf44f5e86a76906931b"), "ID" : "2",
"Name" : "Karan", "Designation" : "CFO", "Skills" : [
"Financial Analysis", "Accounting"], "Salary" : 200000 }
{ "_id" : ObjectId("5d8e4d844f5e86a76906931d"), "ID" : "4", "Name" : "Rohit", "Designation" :
"Senior VP", "Skills" : [
"PHP", ".NET"], "Salary" : 200000 }
{ "_id" : ObjectId("5d8edbba4f5e86a769069320"), "ID" : "6", "Name" : "Gopal", "Designation"
: "COO", "Salary" : 200000 } { "_id" : ObjectId("5d8f06934f5e86a769069323"), "ID" : "9",
"Name" : "Mohit", "Designation" : "CMO", "Skills" : [
"Strategy", "Copywriting"], "Salary" : 200000 }
Display details of employees having ID between 6 and 10

```
> db.employees.find( { ID: {$in: ["6", "7", "8", "9", "10"] } }
).pretty()
{
        "_id" : ObjectId("5d8edbba4f5e86a769069320"),
        "ID" : "6",
        "Name" : "Gopal",
        "Designation" : "COO",
        "Salary" : 200000
} {
        "_id" : ObjectId("5d8ede384f5e86a769069321"),
        "ID" : "7",
        "Name" : "Jay",
        "Designation" : "Software Engineer", "Skills" : [
                "Python",
                "DBMS",
                "Java" ],
        "Salary" : 150000
} {
        "_id" : ObjectId("5d8ede554f5e86a769069322"),
        "ID" : "8",
        "Name" : "Arnab",
        "Designation" : "Software Engineer", "Skills" : [
                "Python",
                "DBMS",
                "Java" ],
        "Salary" : 150000
} {
        "_id" : ObjectId("5d8f06934f5e86a769069323"),
        "ID" : "9",
        "Name" : "Mohit",
        "Designation" : "CMO", "Skills" : [
                "Strategy",
                "Copywriting"
        ],
        "Salary" : 200000
} {
        "_id" : ObjectId("5d8f06934f5e86a769069324"),
        "ID" : "10",
```

```
        "Name" : "Pam",
        "Designation" : "Web Developer",
        "Skills" : [ "HTML",
                "Bootstrap", "PHP",
                "Javascript",
                "React",
                "Java" ],
        "Salary" : 80000
}
```

Display employees having salary between 100000 and 200000

```
> db.employees.find({Salary: {$gte: 100000, $lte:
200000}}).pretty()
{
        "_id" : ObjectId("5d8e4cf44f5e86a76906931b"),
        "ID" : "2",
        "Name" : "Karan",
        "Designation" : "CFO", "Skills" : [
                "Financial Analysis",
                "Accounting" ],
        "Salary" : 200000
} {
        "_id" : ObjectId("5d8e4d844f5e86a76906931c"),
        "ID" : "3",
        "Name" : "Hrushikesh",
        "Designation" : "Product Manager", "Skills" : [
                "Management",
                "SCRUM",
                "Java" ],
        "Salary" : 150000
} {
        "_id" : ObjectId("5d8e4d844f5e86a76906931d"),
        "ID" : "4",
```

```
        "Name" : "Rohit",
        "Designation" : "Senior VP",
        "Skills" : [ "PHP",
                ".NET" ],
        "Salary" : 200000
} {
        "_id" : ObjectId("5d8edba14f5e86a76906931f"),
        "ID" : "5",
        "Name" : "Gaurav",
        "Designation" : "Product Designer",
        "Skills" : [
                "Prototyping",
                "CAD"
        ],
        "Salary" : 150000
} {
        "_id" : ObjectId("5d8edbba4f5e86a769069320"),
        "ID" : "6",
        "Name" : "Gopal",
        "Designation" : "COO",
        "Salary" : 200000
} {
        "_id" : ObjectId("5d8ede384f5e86a769069321"),
        "ID" : "7",
        "Name" : "Jay",
        "Designation" : "Software Engineer", "Skills" : [
                "Python",
                "DBMS",
                "Java" ],
        "Salary" : 150000
} {
        "_id" : ObjectId("5d8ede554f5e86a769069322"),
        "ID" : "8",
        "Name" : "Arnab",
        "Designation" : "Software Engineer", "Skills" : [
                "Python",
                "DBMS",
                "Java" ],
```

```
        "Salary" : 150000
} {
        "_id" : ObjectId("5d8f06934f5e86a769069323"),
        "ID" : "9",
        "Name" : "Mohit",
        "Designation" : "CMO", "Skills" : [
                "Strategy",
                "Copywriting"
        ],
        "Salary" : 200000
}
```


Aggregate

List all the positions in the company

```
> db.employees.aggregate( [ { $group: { _id: "$Designation" } }
] )
{ "_id" : "Software Engineer" }
{ "_id" : "Sales Executive" } { "_id" : "Product
Manager" }
{ "_id" : "Product Designer" }
{ "_id" : "CMO" }
{ "_id" : "Web Developer" }
{ "_id" : "Senior VP" }
{ "_id" : "CFO" } { "_id" : "CEO" }
{ "_id" : "COO" }
```

Regular expressions

List employees whose name starts with ‘A’

```
> db.employees.find({Name: /^A/})
{ "_id" : ObjectId("5d8f06934f5e86a769069325"), "ID" : "11",
"Name" : "Andy", "Designation" : "Software Engineer", "Skills" :
[ "Java", "C++", ".NET" ], "Salary" : 50000 }
```

List employees whose name has the substring ‘it’

```
> db.employees.find({Name: /it/})
```

{ "_id" : ObjectId("5d8e4d844f5e86a76906931d"), "ID" : "4", "Name" : "Rohit", "Designation" : "Senior VP", "Skills" : [
"PHP", ".NET"], "Salary" : 200000 }
{ "_id" : ObjectId("5d8f06934f5e86a769069323"), "ID" : "9",
"Name" : "Mohit", "Designation" : "CMO", "Skills" : [
"Strategy", "Copywriting"], "Salary" : 200000 }
List employees whose name ends with 'm'

```
> db.employees.find({Name: /m$/})
{ "_id" : ObjectId("5d8edba14f5e86a76906931f"), "ID" : "5",
"Name" : "Gaurav", "Designation" : "Product Designer", "Skills"
: [ "Prototyping", "CAD" ], "Salary" : 150000 }
{ "_id" : ObjectId("5d8f06934f5e86a769069324"), "ID" : "10",
"Name" : "Pam", "Designation" : "Web Developer", "Skills" : [
"HTML", "Bootstrap", "PHP", "Javascript", "React", "Java" ],
"Salary" : 80000 }
```

Count

Count total no. of employees

```
> db.employees.count()
13
```

Count the number of employees working as software engineers

```
> db.employees.count({Designation: "Software Engineer"}) 3
```