# **Assignment-I**

Write PL/SQL queries for the following

1) Create tables for - Student(student\_id, first\_name, last\_name, dept, Date\_of\_birth, gender,

religion), Employee, Product, Customer, and Account. Identify relevant attributes for each table

and make sure each table has at least four columns. Ensure each table has a \_ID column e.g.

Employee should have EMPLOYEE\_ID column, Student should have STUDENT\_ID column

etc.

🡺

i) **Student table**

CREATE TABLE Student (

student\_id INT,

first\_name VARCHAR(10),

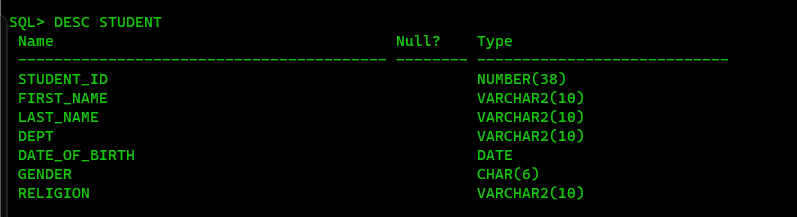
last\_name VARCHAR(10),

dept VARCHAR(10),

Date\_of\_birth DATE,

gender CHAR(6),

religion VARCHAR(10)

);

II)**Employee table**

CREATE TABLE Employee (

EMPLOYEE\_ID NUMBER,

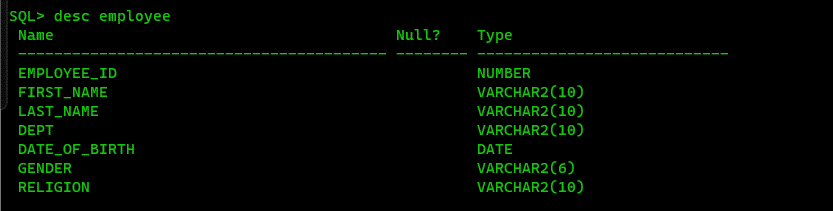
FIRST\_NAME VARCHAR2(10),

LAST\_NAME VARCHAR2(10),

DEPT VARCHAR2(10),

DATE\_OF\_BIRTH DATE,

GENDER VARCHAR2(6),

RELIGION VARCHAR2(10) );

III) Product table

CREATE TABLE Product (

PRODUCT\_ID NUMBER,

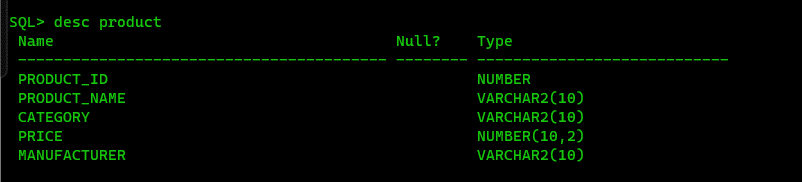
PRODUCT\_NAME VARCHAR2(10),

CATEGORY VARCHAR2(10),

PRICE NUMBER(10, 2),

MANUFACTURER VARCHAR2(10)

);



iv) Customer table

CREATE TABLE Customer (

CUSTOMER\_ID NUMBER ,

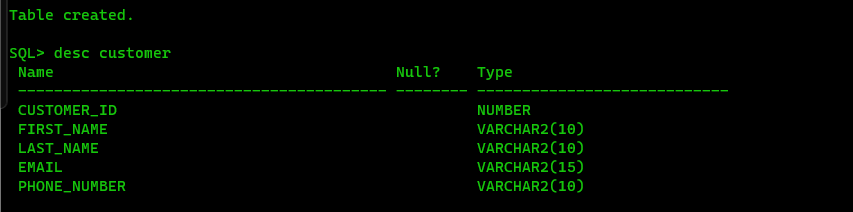
FIRST\_NAME VARCHAR2(10),

LAST\_NAME VARCHAR2(10),

EMAIL VARCHAR2(15),

PHONE\_NUMBER VARCHAR2(10)

);



v) Account table

CREATE TABLE Account (

ACCOUNT\_ID NUMBER,

ACCOUNT\_NUMBER VARCHAR2(10),

ACCOUNT\_TYPE VARCHAR2(10),

BALANCE NUMBER(15, 2),

OPEN\_DATE DATE

);

A screenshot of a computer

Description automatically generated

2. Describe each table.

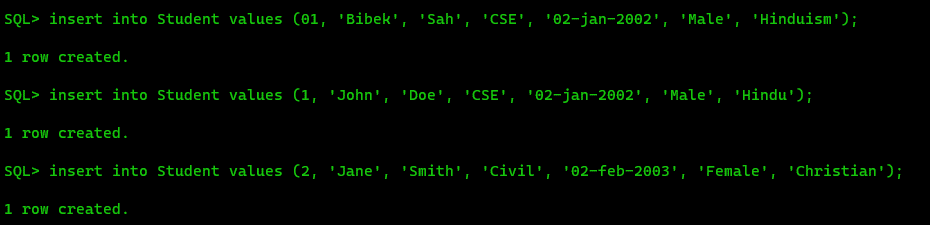
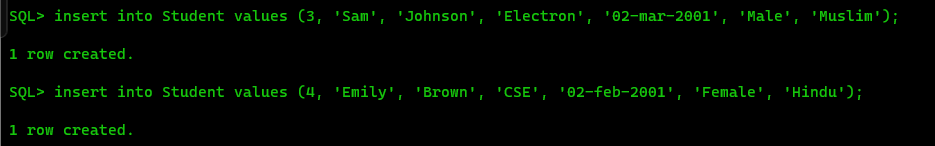
🡺

1. **Student Table:**
   * **STUDENT\_ID:** Unique identifier for each student.
   * **FIRST\_NAME:** First name of the student.
   * **LAST\_NAME:** Last name of the student.
   * **DEPT:** Department to which the student belongs.
   * **DATE\_OF\_BIRTH:** Date of birth of the student.
   * **GENDER:** Gender of the student.
   * **RELIGION:** Religion of the student.
2. **Employee Table:**
   * **EMPLOYEE\_ID:** Unique identifier for each employee.
   * **FIRST\_NAME:** First name of the employee.
   * **LAST\_NAME:** Last name of the employee.
   * **DEPT:** Department to which the employee belongs.
   * **DATE\_OF\_BIRTH:** Date of birth of the employee.
   * **GENDER:** Gender of the employee.
   * **RELIGION:** Religion of the employee.
3. **Product Table:**
   * **PRODUCT\_ID:** Unique identifier for each product.
   * **PRODUCT\_NAME:** Name of the product.
   * **CATEGORY:** Category to which the product belongs.
   * **PRICE:** Price of the product.
   * **MANUFACTURER:** Manufacturer of the product.
4. **Customer Table:**
   * **CUSTOMER\_ID:** Unique identifier for each customer.
   * **FIRST\_NAME:** First name of the customer.
   * **LAST\_NAME:** Last name of the customer.
   * **EMAIL:** Email address of the customer.
   * **PHONE\_NUMBER:** Phone number of the customer.
5. **Account Table:**
   * **ACCOUNT\_ID:** Unique identifier for each account.
   * **ACCOUNT\_NUMBER:** Account number associated with the account.
   * **ACCOUNT\_TYPE:** Type of the account (e.g., savings, checking).
   * **BALANCE:** Current balance in the account.
   * **OPEN\_DATE:** Date when the account was opened.

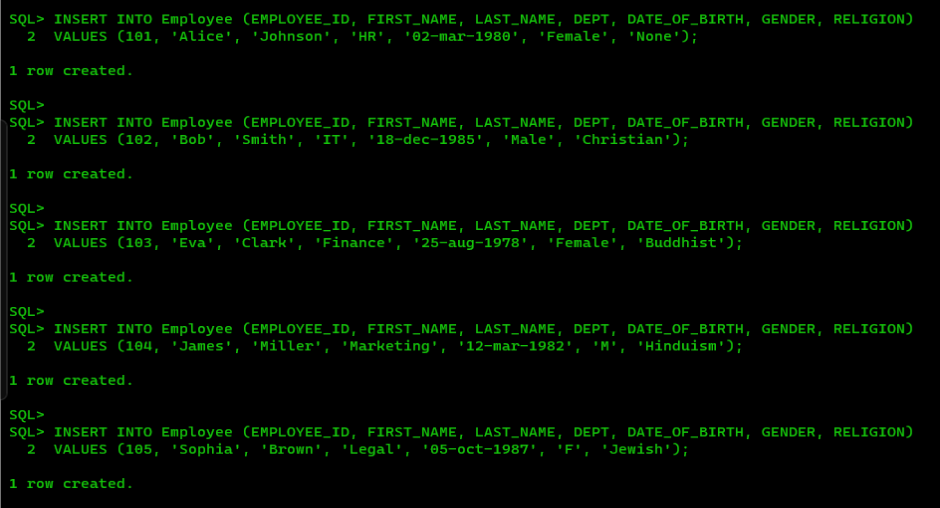
3. Insert at least 5 distinct rows to each table.

🡺 **Student table**

insert into Student values (01, 'Bibek', 'Sah', 'CSE', '02-jan-2002', 'Male', 'Hinduism');



**Employee table**

INSERT INTO Employee (EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, DEPT, DATE\_OF\_BIRTH, GENDER, RELIGION) VALUES (101, 'Alice', 'Johnson', 'HR', '02-mar-1980', 'Female', 'None');

**Product table**

INSERT INTO Product (PRODUCT\_ID, PRODUCT\_NAME, CATEGORY, PRICE, MANUFACTURER) VALUES (501, 'Laptop', 'Electronic', 1200.00, 'HP');

A computer screen shot of green text

Description automatically generated

**Customer table**

INSERT INTO Customer (CUSTOMER\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER) VALUES (1001, 'David', 'Johnson', 'david@em.co', '1234567890');

A computer screen with green text

Description automatically generated

**Account table**

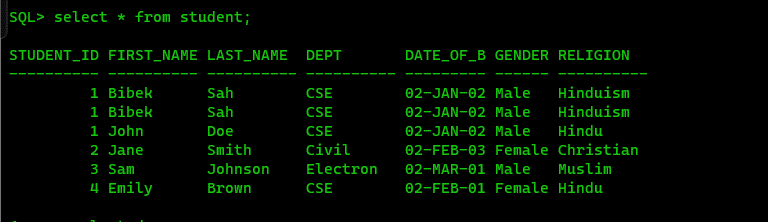
INSERT INTO Account (ACCOUNT\_ID, ACCOUNT\_NUMBER, ACCOUNT\_TYPE, BALANCE, OPEN\_DATE) VALUES (10001, 'A123456', 'Savings', 5000.00, '01-jan-2022');

A screenshot of a computer screen

Description automatically generated

4. Fetch all data from the respective tables.

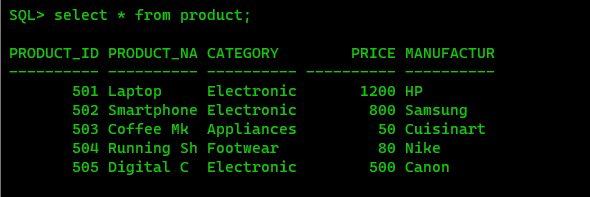
🡺student table:- select \* from student;



Employee table:- select \* from employee;



Product table :- select \* from product;



Customer table :- select \* from customer; A computer screen with green text

Description automatically generated

Account table :- select \* from account;

A screenshot of a computer screen

Description automatically generated

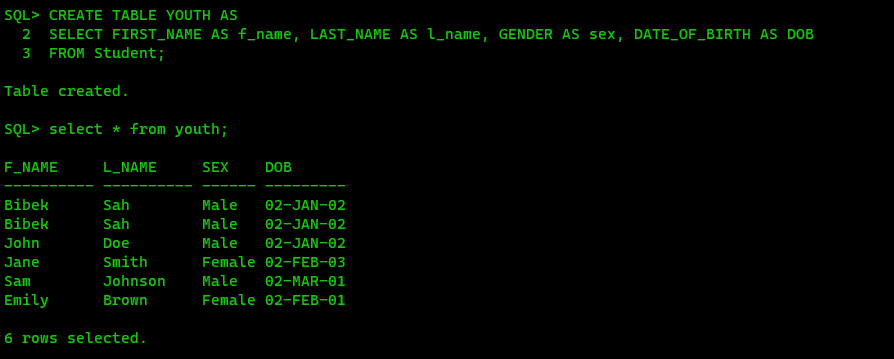
5. Fetch Employee ids and their names from the Employee table.

🡺 select employee\_id, first\_name from employee;



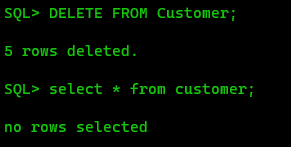
6. Create table YOUTH (f\_name, l\_name, sex, DOB) from the Student table.

🡺 CREATE TABLE YOUTH AS SELECT FIRST\_NAME AS f\_name, LAST\_NAME AS l\_name, GENDER AS sex, DATE\_OF\_BIRTH AS DOB FROM Student;



7. Delete all data from the customer table.

🡺 DELETE FROM Customer;



8. Delete the Account table.

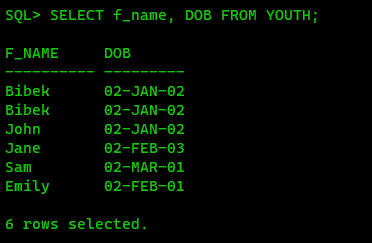
🡺 drop table account;

A screenshot of a computer screen

Description automatically generated

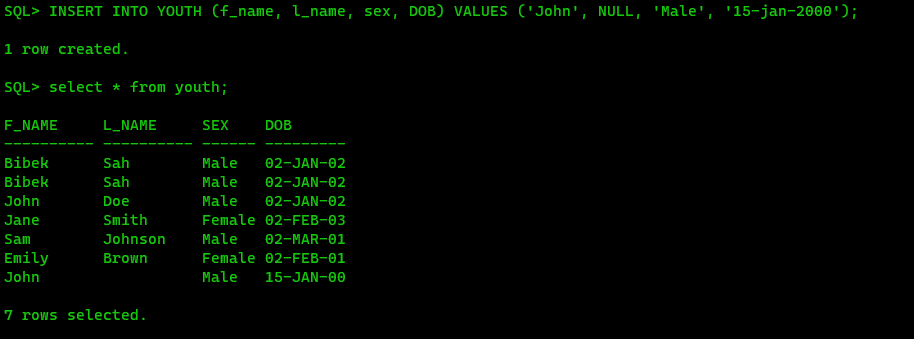
9. Fetch the f\_name and DOB from YOUTH table.

🡺 SELECT f\_name, DOB FROM YOUTH;



10. Insert a new record into the Youth table. And keep NULL value in the l\_name column.

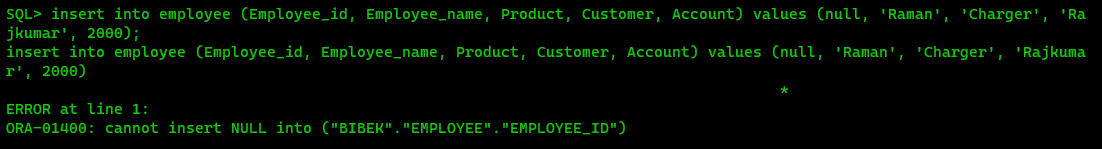
🡺 INSERT INTO YOUTH (f\_name, l\_name, sex, DOB) VALUES ('John', NULL, 'Male', '15-jan-2000');



11. Insert a new record into the Employee table. And keep NULL value in the employee\_id

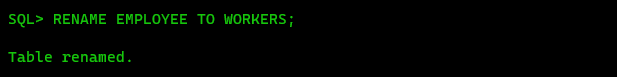
column.

🡺 insert into Employee (Employee\_id, Employee\_name, Product, Customer, Account) values ('null’, ‘Raman’, ‘Charger’, ‘Rajkumar’, 2000);



12. Change the name of the employee table to workers.

🡺RENAME EMPLOYEE TO WORKERS;



13. Increase the size of the dept field in the student table by 10.

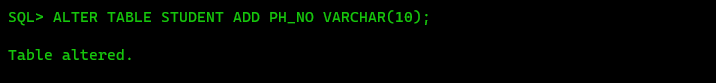
🡺ALTER TABLE STUDENT MODIFY DEPT VARCHAR(10);

A black background with green text

Description automatically generated

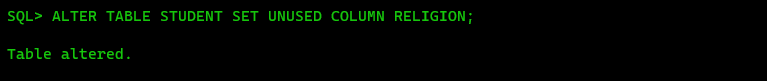
14. Add a column ph\_no in the student table.

🡺 ALTER TABLE STUDENT ADD PH\_NO VARCHAR(10);



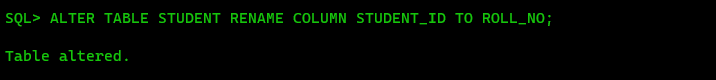
15. Drop the religion attribute from the student table.

🡺 ALTER TABLE STUDENT SET UNUSED COLUMN RELIGION;



16. Rename the student\_id field to roll\_no in the student table.

🡺 ALTER TABLE STUDENT RENAME COLUMN STUDENT\_ID TO ROLL\_NO;



17. Change the datatype and size of the product id column in the product table.

🡺

# **Assignment-II**

Q1: Create a new user making “your\_name” as user-name and “your\_surname” as the password.

🡺create user bibek identified by root;

Q2: Grant all privileges to the newly created user.

🡺 grant all privileges to bibek;

Q3: connect to the new user.

🡺connect

A black screen with green text

Description automatically generated

Q4: Create a table employee with attributes emp\_id, f\_name , l\_name , job\_type, salary, commision, dept, and manager\_id.

🡺 CREATE TABLE Employee (

employee\_id INT,

first\_name VARCHAR(10),

last\_name VARCHAR(10),

job\_types varchar(10),

salary number,

commission number,

dept VARCHAR(10),

manager\_id number);

Q5: Describe the table employee

A screenshot of a computer

Description automatically generated🡺 DESC employee

Q6: Add a new column doj to the employee table.

🡺alter table employee add doj date;

A screenshot of a computer

Description automatically generated

Q7: Create a new table department with attributes d\_name, d\_loc, and hod\_id.

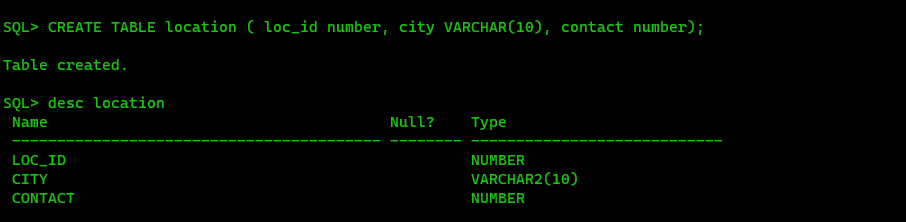
🡺 CREATE TABLE department ( d\_name VARCHAR(10), d\_loc VARCHAR(10), hod\_id number);

A screenshot of a computer

Description automatically generated

Q8: Create another table named location with attributes loc\_id, city and contact\_no.

🡺 CREATE TABLE location ( loc\_id number, city VARCHAR(10), contact number);



Q9. Enhance the size of city attribute in location table by 5.

🡺 ALTER TABLE location MODIFY city VARCHAR(5);

A screenshot of a computer

Description automatically generated

Q10. Delete the contact\_no attribute in the location table.

🡺 ALTER TABLE location SET UNUSED COLUMN contact;

A screenshot of a computer

Description automatically generated

Q11. Rename the city attribute in the location table to address.

🡺 ALTER TABLE location RENAME COLUMN city TO address;

A screenshot of a computer

Description automatically generated

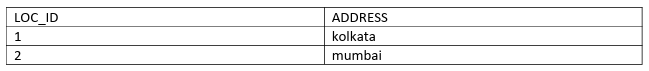
Q12. Change the name of the table from location to loc.

🡺 RENAME location To loc;

A screen shot of a computer

Description automatically generated

Q13. Insert the following values into the loc table.



🡺 insert into loc values (01, 'Kol');

A black screen with green text

Description automatically generated

Q14. Show the values of location table.

🡺 select \* from loc;

A screenshot of a computer

Description automatically generated

Q15. Delete all values and spaces consumed by loc table.

🡺 TRUNCATE TABLE loc;

A screen shot of a computer

Description automatically generated

Q16. Delete the loc table.

🡺 drop table loc;

A black screen with green text

Description automatically generated

Q17. Insert the following values into the department table.

A blank list of days

Description automatically generated with medium confidence

🡺 insert into department values ('&d\_name', '&d\_loc', &hod\_id);

A computer screen with green text

Description automatically generated

Q18. Insert the following values into the employee table.

A table of numbers and letters

Description automatically generated with medium confidence

🡺 insert into employee values (&employee\_id, '&first\_name', '&last\_name', '&job\_type', &salary, &commission, '&dept', &manager\_id, '&doj');

A computer screen with green text

Description automatically generated

Q19. Save the database.

🡺commit;

A black screen with green text

Description automatically generated

Q20: Show all the attribute values of the department table.

🡺 select \* from department;

A screenshot of a computer program

Description automatically generated

Q21: Display the department names and their locations.

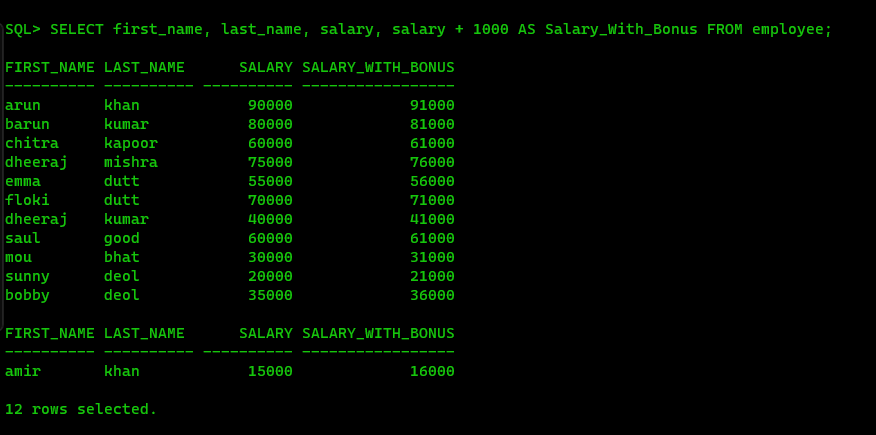
🡺 select d\_name, d\_loc from department;

A screenshot of a computer

Description automatically generated

Q22: Show the employee’s first name, last name, current salary and the salary with a 1000 rupees bonus.

🡺 SELECT first\_name, last\_name, salary, salary + 1000 AS Salary\_With\_Bonus FROM employee;



Q23: Show the employee’s annual salary with a 1000 rupees yearly bonus and the annual salary with a 100 rupees monthly bonus.

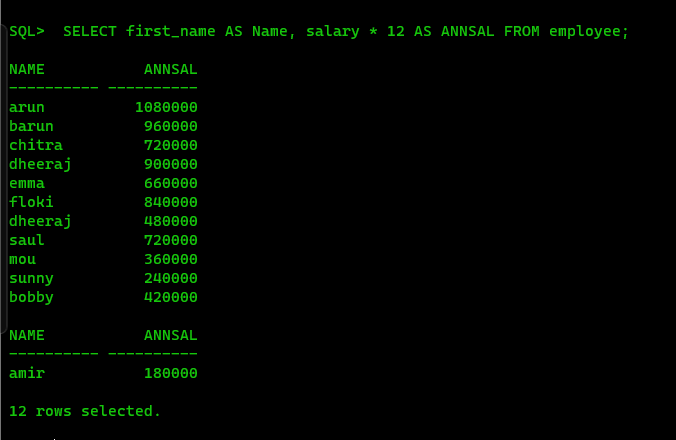
🡺 SELECT first\_name, last\_name, salary \* 12 + 1000 AS Yearly\_Bonus, (salary + 100) \* 12 AS Monthly\_Bonus FROM employee;

A screenshot of a computer screen

Description automatically generated

Q24: Show f\_name as Name and annual salary as ANNSAL from the employee table.

🡺 SELECT first\_name AS Name, salary \* 12 AS ANNSAL FROM employee;



Q25: Show the L\_name as SurName and 100 rupees incremented salary as NewSal from the employee table.

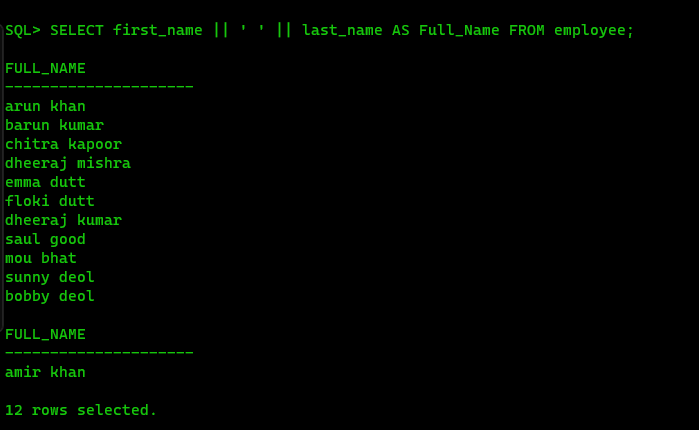
🡺 SELECT last\_name AS SurName, salary + 100 AS NewSal FROM employee;

A screenshot of a computer

Description automatically generated

Q26: Display the employees f\_name and l\_name joined together using the concatenation operator.

🡺 SELECT first\_name || ' ' || last\_name AS Full\_Name FROM employee;



Q27: Show the f\_name, l\_name and job\_type as Employees.

🡺 SELECT first\_name, last\_name, job\_types AS Employees FROM employee;

A screenshot of a computer

Description automatically generated

Q28: Show the employee details in the following fassion:

Employees Details

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

arun khan is a manager

barun kumar is a manager

……..

……..

🡺 SELECT first\_name || ' ' || last\_name || ' is a ' || job\_types AS Employees\_Details FROM employee;

A screenshot of a computer

Description automatically generated

Q29: Show the monthly salary details in the following fassion:

Monthly Salary Details

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

arun's monthly salary is 90000

………

🡺 SELECT first\_name || '''s monthly salary is ' || TO\_CHAR(salary) AS Monthly\_Salary\_Detalil FROM employee;

A screenshot of a computer

Description automatically generated

Q30: Show the department names from the employee table.

🡺 select dept from employee;

A screenshot of a computer

Description automatically generated

Q31: Show the distinct department names from the employee table.

🡺 SELECT DISTINCT dept FROM employee;

A screen shot of a computer

Description automatically generated

Q32: Show the employees earning more than 50000.

🡺 SELECT \* FROM employee WHERE salary > 50000;

A screenshot of a computer

Description automatically generated

Q33. Show the employee’s id’s who are not working under manager id-1.

🡺 SELECT employee\_id FROM employee WHERE manager\_id != 1;

A screen shot of a computer

Description automatically generated

Q34: Show the employee’s names and salaries whose salary ranges between 40000 to 70000.

🡺 SELECT first\_name, last\_name, salary FROM employee WHERE salary BETWEEN 40000 AND 70000;

A black screen with green text

Description automatically generated

Q35: Show the employees who work for manager id 1 or 6 or 8.

🡺 SELECT \* FROM employee WHERE manager\_id IN (1, 6, 8);

A screenshot of a computer screen

Description automatically generated

Q36: Select the first names and salaries of those employee whose last name is khan.

🡺 SELECT first\_name, salary FROM employee WHERE last\_name = 'khan';

A black screen with green text

Description automatically generated

Q37: Select the first names and salaries of those employee whose last name starts with k.

🡺 SELECT first\_name, salary FROM employee WHERE last\_name LIKE 'k%';

A screen shot of a computer

Description automatically generated

Q38: Select the first name, last name and salary of those employee whose last name starts with k and ends with r.

🡺 SELECT first\_name, last\_name, salary FROM employee WHERE last\_name LIKE 'k%r';

A black screen with green text

Description automatically generated

Q39: Select the employees whose 3rd letter of their last name is o.

🡺 SELECT \* FROM employee WHERE SUBSTR(last\_name, 3, 1) = 'o';

A screenshot of a computer

Description automatically generated

Q40: Select the employees who are not working under any manager.

🡺 SELECT \* FROM employee WHERE manager\_id IS NULL;

A screenshot of a computer screen

Description automatically generated

Q41: Select the employees who work as engineers with salary greater than 50000.

🡺 SELECT \* FROM employee WHERE job\_types = 'engineer' AND salary > 50000;

A screenshot of a computer

Description automatically generated

Q42: Select the employees who work in the production department or earns more than 60000.

🡺 SELECT \* FROM employee WHERE dept = 'production' OR salary > 60000;

A screenshot of a computer screen

Description automatically generated

Q43: Select those employees who are not managers or engineers or clerks.

🡺 SELECT \* FROM employee WHERE job\_types NOT IN ('manager', 'engineer', 'clerk');

A screenshot of a computer

Description automatically generated

Q44: Select the employees who earns more than 49000 or less than 29000.

🡺 SELECT \* FROM employee WHERE salary > 49000 OR salary < 29000;

A screenshot of a computer

Description automatically generated

Q45. Select the employees who don’t have an ‘o’ as the 2nd last letter of their last name.

🡺 SELECT \* FROM employee WHERE SUBSTR(last\_name, -2, 1) != 'o';

A screenshot of a computer

Description automatically generated

Q46. Select the employees who get commission.

🡺 SELECT \* FROM employee WHERE commission IS NOT NULL;

A screenshot of a computer

Description automatically generated

Q47. WAQ to display the current date.

🡺 SELECT SYSDATE AS current\_date FROM dual;

A black screen with green text

Description automatically generated

Q48. Show the total experience in weeks for all the employees.

🡺 SELECT employee\_id, first\_name, last\_name, doj, TRUNC((SYSDATE - doj) / 7) AS experience\_in\_weeks FROM employee;

A screenshot of a computer screen

Description automatically generated

Q49. Find the employees working under employee\_id 2.

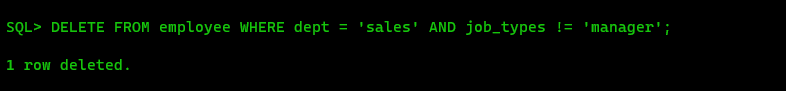
🡺 SELECT \* FROM employee WHERE employee\_id = 2;

A screenshot of a computer

Description automatically generated

Q50. Delete the employees from sales department if they are not working as managers.

🡺 DELETE FROM employee WHERE dept = 'sales' AND job\_types != 'manager';



Q51. Insert the following two rows in the employee table without inserting any value in the department field.

A number of numbers on a white background

Description automatically generated

🡺 insert into employee values (&employee\_id, '&first\_name', '&last\_name', '&job\_type', &salary, &commission, '&dept', &manager\_id, '&doj');

A computer screen shot of green text

Description automatically generated

Q52. . Insert the following two rows in the department table.

A group of black text

Description automatically generated

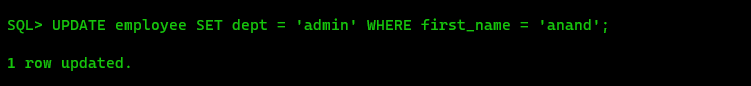
🡺 insert into department values ('&d\_name', '&d\_loc', &hod\_id);

A screenshot of a computer program

Description automatically generated

Q53. Update the employee table. Assign Anand to the admin department.

🡺 UPDATE employee SET dept = 'admin' WHERE first\_name = 'anand';



Q54. Update the manager\_id from 2 to 1 in the employee table.

🡺 UPDATE employee SET manager\_id = 1 WHERE manager\_id = 2;

A black screen with green text

Description automatically generated

Q55. Display the employee details in descending order on their salary.

🡺 SELECT \* FROM employee ORDER BY salary DESC;

A screenshot of a computer

Description automatically generated

Q56. Display the employee details in ascending order on their l\_name.

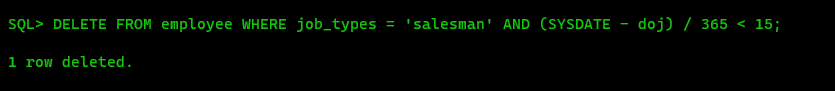
🡺 SELECT \* FROM employee ORDER BY last\_name ASC;

A screenshot of a computer screen

Description automatically generated

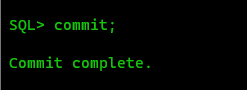
Q57. Delete the employees who are working as salesmen and having less experience than 15 years.

🡺 DELETE FROM employee WHERE job\_types = 'salesman' AND (SYSDATE - doj) / 365 < 15;



Q58. Commit the database.

🡺commit;



# **Assignment-III**

Name :- Bibek Chand Sah

Roll No. :- 22054029

Section :- CSE-05

1. Create a tabme emp12, with attributes empid, empname, phno, age, and emp\_country. Make emp id the primary key of the table. Make empname not null, phno unique and put a check constraint on age which make sure employees are not below 18. Put a default value “India” on emp\_country.

🡺 CREATE TABLE emp12 (empid INT CONSTRAINT empid\_pk PRIMARY KEY, empname VARCHAR(10) CONSTRAINT empname\_nn NOT NULL, phno VARCHAR(10) CONSTRAINT phno\_uk UNIQUE, age INT CONSTRAINT age\_ck CHECK (age >= 18), emp\_country VARCHAR(10) DEFAULT 'India' );

A screen shot of a computer

Description automatically generated

2. Insert the following tuples into emp12 table.

A table with numbers and names

Description automatically generated

🡺 INSERT INTO emp12 VALUES (&empid, '&empname', '&phno', &age, '&emp\_country');

ALTER TABLE emp12 DROP CONSTRAINT empname\_nn;

ALTER TABLE emp12 DROP CONSTRAINT phno\_uk;

ALTER TABLE emp12 DROP CONSTRAINT empid\_pk;

ALTER TABLE emp12 DROP CONSTRAINT age\_ck;

A computer screen with green text

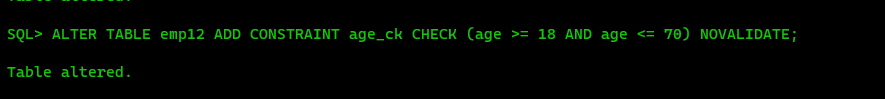
Description automatically generated

3. Drop the check constraint on age.

🡺 ALTER TABLE emp12 DROP CONSTRAINT age\_ck;

4. Put a default constraint with value 18 on age along with a check constraint which ensures age value is greater than or equal to 18 and less than or equal to 70.

🡺 ALTER TABLE emp12 ADD CONSTRAINT age\_ck CHECK (age >= 18 AND age <= 70) NOVALIDATE;



5. Insert the following rows in the emp12 table.

A close-up of a table

Description automatically generated🡺 INSERT INTO emp12 VALUES (&empid, '&empname', '&phno', &age, '&emp\_country');

A computer screen with green text

Description automatically generated

6. Create table projectx with attributes pid, pname, phead, ploc and pmembers. pmembers should have a default value of 5. Make (pid,pname) the primary key. Use a constraint named uc\_px\_34 to make phead and ploc unique. Put a check constraint named ck\_px\_5 on pmembers which ensures that the number of pmembers must not exceed 5.

🡺 CREATE TABLE projectx ( pid INT, pname VARCHAR(50), phead VARCHAR(50), ploc VARCHAR(50), pmembers INT DEFAULT 5, CONSTRAINT pk\_projectx PRIMARY KEY (pid, pname), CONSTRAINT uc\_px\_34 UNIQUE (phead, ploc), CONSTRAINT ck\_px\_5 CHECK (pmembers <= 5) );

A screen shot of a computer

Description automatically generated

7. Drop the primary key of the projectx table.

🡺 ALTER TABLE projectx DROP CONSTRAINT pk\_projectx;

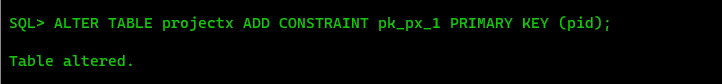
A black background with green text

Description automatically generated

8. Put a constraint named pk\_px\_1 on pid, making pid the primary key of the table.

🡺 ALTER TABLE project DROP CONSTRAINT pk\_projectx;

ALTER TABLE projectx ADD CONSTRAINT pk\_px\_1 PRIMARY KEY (pid);



9. Insert the following rows in projectx table.

A table with numbers and letters

Description automatically generated

🡺 ALTER TABLE projectx MODIFY pid varchar2(10);

INSERT INTO projectx VALUES ('&pid', '&pname', '&phead', '&ploc', &pmembers);

A computer screen with green text

Description automatically generated

10. Delete all rows from emp12 table except the rows having primary keys (112, 113, 114).

🡺 DELETE FROM emp12 WHERE empid NOT IN (112, 113, 114);

A black screen with green text

Description automatically generated

11. Insert a colunm named project in the emp12 table.

🡺 ALTER TABLE emp12 ADD project VARCHAR(10);

A black background with green text

Description automatically generated

12. Update the table as following.

A table with numbers and text

Description automatically generated

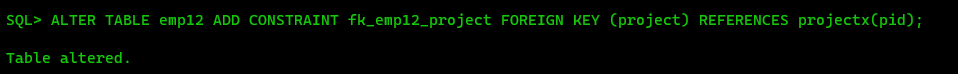
🡺 INSERT INTO emp12 VALUES (&empid, '&empname', '&phno', &age, '&emp\_country', '&project');

A screen shot of a computer

Description automatically generated

13. Make the project attribute of the emp12 table a foreign key that links it to the pid attribute of the projectx table.

🡺 ALTER TABLE emp12 ADD CONSTRAINT fk\_emp12\_project FOREIGN KEY (project) REFERENCES projectx(pid);



14. Insert the following tuples in the emp12 table.

A table with numbers and letters

Description automatically generated

🡺 INSERT INTO emp12 VALUES (&empid, '&empname', '&phno', &age, '&emp\_country', '&project');

ALTER TABLE emp12 DROP CONSTRAINT FK\_EMP12\_PROJECT;

A black screen with green text

Description automatically generated

15. Make the phead attribute of the projectx table a foreign key that links it to the empid attribute of the emp12 table.

🡺 ALTER TABLE project ADD CONSTRAINT fk\_projectx\_emp12 FOREIGN KEY (phead) REFERENCES emp12(empid);

ADD CONSTRAINT CFK FOREIGN KEY(PHEAD) REFERENCES EMP12(EMPID);

ALTER TABLE EMP12 MODIFY EMPID NUMBER PRIMARY KEY

Relation can’t be established until there are duplicate EMPIDs. Hence,

Delete duplicate empids (I.e Lily)

DELETE FROM EMP12 WHERE EMPNAME = 'Lily';

Make empid primary key

ALTER TABLE EMP12 MODIFY EMPID NUMBER PRIMARY KEY;

Create relation :

ALTER TABLE project ADD CONSTRAINT fk\_phead\_emp12 FOREIGN KEY (phead) REFERENCES emp12(empid);

16. Display all the constraints there are in emp12 table.

🡺 SELECT constraint\_name, constraint\_type, table\_name FROM all\_constraints WHERE table\_name = 'EMP12';

17. Drop all the constraints in emp12 table one by one.

🡺 ALTER TABLE emp12 DROP CONSTRAINT pk\_px\_1;

ALTER TABLE emp12 DROP CONSTRAINT uc\_px\_34;

ALTER TABLE emp12 DROP CONSTRAINT ck\_px\_5;

18. Make emp\_id the primary key of the employee table (of assignment 2).

🡺 ALTER TABLE employee ADD CONSTRAINT pk\_employee PRIMARY KEY (emp\_id);

19. Make the dept attribute of the employee table a foreign key refering to the department table (of assignment 2).

🡺 ALTER TABLE employee ADD CONSTRAINT fk\_employee\_department FOREIGN KEY (dept) REFERENCES department(dept\_id);

20. Make d\_name the primary key of the department table (of assignment 2).

🡺 ALTER TABLE department ADD CONSTRAINT pk\_department PRIMARY KEY (d\_name);

21. Make the dept attribute of the employee table a foreign key refering to the d\_name attribute of the department table (of assignment 2)

🡺 ALTER TABLE employee ADD CONSTRAINT fk\_employee\_department FOREIGN KEY (dept) REFERENCES department(d\_name);

# **Assignment-IV**

Name :- Bibek Chand Sah

Roll No. :- 22054029

Section :- CSE-05

1. WAQ to display the current date.

🡺 SELECT SYSDATE AS current\_date FROM dual;

A black screen with green text

Description automatically generated

2. Assume that employees serve a six month provisional period starting from their DOJ. After 6 months they get permanent status. Display all the employees’ names, doj and the date on which they received their permanent status as permanent\_date.

🡺 SELECT f\_name, l\_name, doj, doj + INTERVAL '6' MONTH AS permanent\_date FROM employee;

3. WAQ to display the last date of this current month.

🡺 SELECT LAST\_DAY(SYSDATE) AS last\_date\_of\_month FROM dual;

A black screen with green text

Description automatically generated

4. For all the employees display their emp\_id, f\_name and their total experience in months.

🡺 SELECT employee\_id, first\_name, last\_name, MONTHS\_BETWEEN(SYSDATE, doj) AS total\_experience\_months FROM employee;

A screenshot of a computer screen

Description automatically generated

5. WAQ to display the date of next TUESDAY.

🡺 SELECT NEXT\_DAY(SYSDATE, 'TUESDAY') AS next\_tuesday\_date FROM dual;

A screen shot of a black background

Description automatically generated

6. WAQ to extract the current month.

🡺 SELECT EXTRACT(MONTH FROM SYSDATE) AS current\_month FROM dual;

A screen shot of a computer

Description automatically generated

7. WAQ to extract the current year.

🡺 SELECT EXTRACT(YEAR FROM SYSDATE) AS current\_year FROM dual;

A black screen with green text

Description automatically generated

8. WAQ to display the absolute value of -505.

🡺 SELECT ABS(-505) AS absolute\_value FROM dual;

A black screen with green text

Description automatically generated

9. WAQ to display the ceiling of 10.44 and 10.50 and 10.65.

🡺 SELECT CEIL(10.44) AS ceil\_1, CEIL(10.50) AS ceil\_2, CEIL(10.65) AS ceil\_3 FROM dual;

A black screen with green text

Description automatically generated

10. WAQ to display the floor value of 10.44 and 10.50 and 10.65.

🡺 SELECT FLOOR(10.44) AS floor\_1, FLOOR(10.50) AS floor\_2, FLOOR(10.65) AS floor\_3 FROM dual;

A black screen with green text

Description automatically generated

11. Find the logarithmic value of 10 base 2.

🡺 SELECT LOG(10, 2) AS logarithmic\_value FROM dual;

A black screen with green text

Description automatically generated

12. Display the remainder in 594/7.

🡺 SELECT MOD(594, 7) AS remainder FROM dual;

A black screen with green text

Description automatically generated

13. WAQ to display the value of 8 to the power 3.

🡺 SELECT POWER(8, 3) AS power\_value FROM dual;

A black screen with green text

Description automatically generated

14. WAQ to display the square root of 3481.

🡺 SELECT SQRT(3481) AS square\_root FROM dual;

A black screen with green text

Description automatically generated

15. Display the following rounding operations- round(45.923,2), round(45.923,0), round(45.923,-1), round(45.923,2), round(45.923,0), round(45.923,-2).

🡺 SELECT ROUND(45.923, 2) AS round\_1, ROUND(45.923, 0) AS round\_2, ROUND(45.923, -1) AS round\_3, ROUND(45.923, -2) AS round\_4 FROM dual;

A black screen with green text

Description automatically generated

16. Display the following rounding operations- trunc(45.888,2), trunc(56.758,0), trunc(49.245,-2), trunc(45.888,2), round(45.888,2).

🡺 SELECT TRUNC(45.888, 2) AS trunc\_1, TRUNC(56.758, 0) AS trunc\_2, TRUNC(49.245, -2) AS trunc\_3 FROM dual;

A black screen with green text

Description automatically generated

17. WAQ to return the sign of 20 and -67.60 and 0.

🡺 SELECT SIGN(20) AS sign\_1, SIGN(-67.60) AS sign\_2, SIGN(0) AS sign\_3 FROM dual;

A black screen with green text

Description automatically generated

18. Display the value of cos(45), sin(45), and tan(45).

🡺 SELECT COS(45) AS cosine, SIN(45) AS sine, TAN(45) AS tangent FROM dual;

A black screen with green text

Description automatically generated

19. Display the ASCII character corresponding to the integer 79.

🡺 SELECT CHR(79) AS ascii\_character FROM dual;

A screen shot of a computer

Description automatically generated

20.Display the f\_name and l\_name together using the concat() function.

🡺 SELECT CONCAT ('Bibek', 'Sah') FROM DUAL;

A black screen with green text

Description automatically generated

21. Display all the f\_names in capital letters.

🡺 SELECT UPPER(first\_name) AS capital\_f\_name FROM employee;

A screenshot of a computer

Description automatically generated

22. Find the length of the first name and last name of all the employees who work in the sales department.

🡺 SELECT LENGTH(first\_name) AS first\_name\_length, LENGTH(last\_name) AS last\_name\_length FROM employee WHERE dept = 'sales';

A black screen with green text

Description automatically generated

23. Determine the tax-rate for each employee based on their monthly salary. The tax-rates are

as per the following table. If salary is < 2000 then tax rate is 0 %, 20000<=salary<40000 9%, 40000<=salary<60000 20%, 60000<=salary<80000 30%, salary>=80000 45%

A white rectangular object with black text

Description automatically generated with medium confidence

🡺 SELECT employee\_id, first\_name, last\_name, salary, CASE WHEN salary < 2000 THEN 0 WHEN salary >= 20000 AND salary < 40000 THEN 9 WHEN salary >= 40000 AND salary < 60000 THEN 20 WHEN salary >= 60000 AND salary < 80000 THEN 30 WHEN salary >= 80000 THEN 45 END AS tax\_rate FROM employee;

A screenshot of a computer screen

Description automatically generated

24. Find the average salary, maximum salary, minimum salary and the sum of salaries from the employee table.

🡺 SELECT AVG(salary) AS average\_salary, MAX(salary) AS maximum\_salary, MIN(salary) AS minimum\_salary, SUM(salary) AS total\_salary FROM employee;

A screenshot of a computer screen

Description automatically generated

25. Find the average salary, maximum salary, minimum salary and the sum of salaries of the

employees who work for the sales department.

🡺 SELECT AVG(salary) AS average\_salary, MAX(salary) AS maximum\_salary, MIN(salary) AS minimum\_salary, SUM(salary) AS total\_salary FROM employee WHERE dept = 'sales';

A screenshot of a computer screen

Description automatically generated

26. Find the newest and oldest employee.

🡺 SELECT MAX(doj) AS newest\_employee, MIN(doj) AS oldest\_employee FROM employee;

A black screen with green text

Description automatically generated

27. Find those two employees whose l\_name comes first and last in alphabetical order.

🡺 SELECT MIN(last\_name) AS first\_employee, MAX(last\_name) AS last\_employee FROM employee;

A screen shot of a computer

Description automatically generated

28. Find the number of engineers.

🡺 SELECT COUNT(\*) AS num\_engineers FROM employee WHERE job\_types = 'engineer';

A black screen with green text

Description automatically generated

29. Find the number of departments from the employee table.

🡺 SELECT COUNT(DISTINCT dept) AS num\_departments FROM employee;

A computer screen with green text

Description automatically generated

30. Find the average commission from the employee table.

🡺 SELECT AVG(commission) AS average\_commission FROM employee;

A screen shot of a computer

Description automatically generated

1. Find the date of next Sunday.

🡺 select next\_day(sysdate,'sunday') from dual;

A black screen with green text

Description automatically generated

1. Find 7 raised to the power 7.

🡺select power(7,7) from dual;

A black screen with green text

Description automatically generated

1. Input your name(first\_name last\_name) in upper case “JAYANTA MONDAL”. The output should come under the alias MY\_NAME and the name should appear in lower\_case but the first letter of first\_name and last\_name should be in upper\_case.

è SELECT INITCAP('JAYANTA MONDAL') AS NAME FROM DUAL;

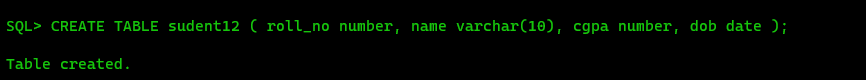
A black screen with green text

Description automatically generated

1. Create and insert the following:       STUDENT12

|  |  |  |  |
| --- | --- | --- | --- |
| Roll\_No | Name | CGPA | DOB |
| 120 | Avi | 6.55 | 09-Jan-2000 |
| 134 | Beena | 8.20 | 31-Aug-2002 |
| 155 | Charu | 8.20 | 23-Dec-1999 |
| 202 | Dawar | 9.15 | 09-Jan-2000 |

🡺 CREATE TABLE student12 ( roll\_no number, name varchar(10), cgpa number, dob date );



INSERT INTO student12 VALUES (&roll\_no, '&name', &cpga, '&dob')

A black screen with green text

Description automatically generated

1. Say, stipend is given to all students. 10000 if cgpa>9, 7000 if cgpa>8, 5000 if cgpa>7, rest are given 1000 rupees. List the students name and stipend received by them.

🡺 select Roll\_no,name,CGPA,decode(trunc(CGPA,0),9,10000,8,7000,7,5000,1000) as stipend from student12;

A screen shot of a computer

Description automatically generated

1. Show the students name in upper\_case, and DOB in the following format : five-february-two thousand twenty-four.

è select upper(name), to\_char(dob,'ddsp-month-yyyysp') dob from student12;

A screenshot of a computer program

Description automatically generated

1. Count the number of students present in the student12 table.

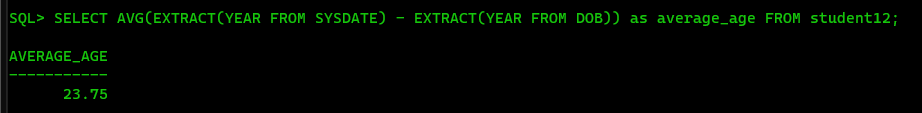
🡺 SQL> select count(\*) as student\_count from student12;

A black screen with green text

Description automatically generated

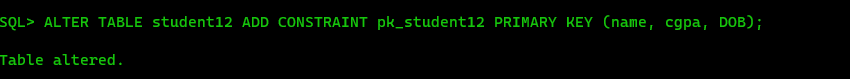
1. Select the average age of the students.

🡺 SELECT AVG(EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM DOB)) as average\_age FROM student12;



1. Make the set of name,cgpa and DOB, the primary key of the student12 table.

🡺 ALTER TABLE student12 ADD CONSTRAINT pk\_student12 PRIMARY KEY (name, cgpa, DOB);



1. Make roll\_no the new primary key of the table.

🡺 ALTER TABLE student12 DROP CONSTRAINT pk\_student12;

ALTER TABLE student12 ADD CONSTRAINT pk\_student12 PRIMARY KEY (roll\_no);

A computer screen with green text

Description automatically generated

# **Assignment-V**

Name :- Bibek Chand Sah

Roll :- 22054029

Branch :- CSE

Section :- CSE-05

1. Find the average salary of each department.

🡺 SELECT dept, AVG(salary) AS average\_salary FROM employee GROUP BY dept;

A screenshot of a computer screen

Description automatically generated

2. Find the average salary for each jobtype according to each department.

🡺 SELECT dept, job\_types, AVG(salary) AS average\_salary FROM employee GROUP BY dept, job\_types;

A screenshot of a computer screen

Description automatically generated

3. Find the department names and their corresponding average salary where the average

salary is greater than 40000.

🡺 SELECT dept, AVG(salary) AS average\_salary FROM employee GROUP BY dept HAVING AVG(salary) > 40000;

A screen shot of a black background

Description automatically generated

4. Select the departments where the maximum salary is more than 55000.

🡺 SELECT dept FROM employee GROUP BY dept HAVING MAX(salary) > 55000;

A black screen with green text

Description automatically generated

5. Find the department names and their average salary where the maximum salary of the

department is higher than 55000.

🡺 SELECT dept, AVG(salary) AS average\_salary FROM employee GROUP BY dept HAVING MAX(salary) > 55000;

A black screen with green text

Description automatically generated

6. Display the job\_types and the total monthly salary for each jobtypes as “PAYROLL”, where

the total payroll according to jobtypes exceeds 100000/month.

🡺 SELECT job\_types, SUM(salary) AS total\_salary FROM employee GROUP BY job\_types HAVING SUM(salary) > 100000;

A black screen with green text

Description automatically generated

7. Display the job\_types and the total monthly salary for each jobtypes as “PAYROLL”, where

the total payroll according to jobtypes exceeds 100000/month and jobtype is not engineer.

🡺 SELECT job\_types, SUM(salary) AS total\_salary FROM employee WHERE job\_types != 'engineer' GROUP BY job\_types HAVING SUM(salary) > 100000;

A black screen with green text

Description automatically generated

8. Display the job\_types and the total monthly salary for each jobtypes as “PAYROLL”, where

the total payroll according to jobtypes exceeds 60000/month and jobtype is not engineer

and sort the list in ascending order of sum of salary.

🡺 SELECT job\_types, SUM(salary) AS total\_salary FROM employee WHERE job\_types != 'engineer' GROUP BY job\_types HAVING SUM(salary) > 60000 ORDER BY SUM(salary) ASC;

A black screen with green text

Description automatically generated

9. Display the job\_types and the total monthly salary for each jobtypes as “PAYROLL”, where

the total payroll according to jobtypes exceeds 50000/month and jobtype is not engineer

and sort the list in descending order of sum of salary.

🡺 SELECT job\_types, SUM(salary) AS total\_salary FROM employee WHERE job\_types != 'engineer' GROUP BY job\_types HAVING SUM(salary) > 50000 ORDER BY SUM(salary) DESC;

A black screen with green text

Description automatically generated

10. Find the maximum average salary according to departments.

🡺 SELECT DEPT, AVG\_SALARY AS MAX\_AVG\_SALARY FROM (SELECT DEPT, AVG(SALARY) AS AVG\_SALARY FROM EMPLOYEE GROUP BY DEPT) Subquery WHERE AVG\_SALARY = (SELECT MAX(AVG\_SALARY) FROM (SELECT AVG(SALARY) AS AVG\_SALARY FROM EMPLOYEE GROUP BY DEPT ));

A screen shot of a computer

Description automatically generated

11. Find the minimum average salary according to jobtypes.

🡺 SELECT MIN(AVG(SALARY)) AS MIN\_AVG\_SALARY FROM EMPLOYEE GROUP BY JOB\_TYPES;

A black screen with green text

Description automatically generated

12. Find the employee name and date of joining who are working in delhi.

🡺 SELECT e.FIRST\_NAME || ' ' || e.LAST\_NAME AS EMPLOYEE\_NAME, e.DOJ FROM EMPLOYEE e JOIN DEPARTMENT d ON e.DEPT = d.D\_NAME WHERE d.D\_LOC = 'delhi';

A black screen with green text

Description automatically generated

13. Create the table ‘Emp\_Address’ for storing the permanent address of the employees and

insert the values.

A table with names of cities

Description automatically generated

🡺 CREATE TABLE Emp\_Address (emp\_id INT, city VARCHAR(15), district VARCHAR(15), state VARCHAR(15));

INSERT INTO Emp\_Address VALUES (&emp\_id, '&city', '&district', '&state');

A screen shot of a computer

Description automatically generated

14. Display the employee name with their home city and the city they work in.

🡺 SELECT e.FIRST\_NAME || ' ' || e.LAST\_NAME AS Employee\_Name, a\_home.City AS Home\_City, d.D\_LOC AS Working\_City FROM EMPLOYEE e JOIN EMP\_ADDRESS a\_home ON e.EMP\_ID = a\_home.EMP\_ID JOIN DEPARTMENT d ON e.DEPT = d.D\_NAME;

15. Create the following Job\_Grades table.

A screenshot of a computer

Description automatically generated

🡺 CREATE TABLE JOB\_GRADES (GRADE VARCHAR2(5), LOWEST\_SAL NUMBER(10), HIGHEST\_SAL NUMBER(10));

INSERT INTO JOB\_GRADES VALUES('A',10000,24999);

INSERT INTO JOB\_GRADES VALUES('B',25000,49999);

INSERT INTO JOB\_GRADES VALUES('C',50000,100000);

A computer screen with green text

Description automatically generated

16. Display the employee names along with their salary and job\_grade.

🡺 SELECT e.FIRST\_NAME || ' ' || e.LAST\_NAME AS EMPLOYEE\_NAME, e.SALARY, j.GRADE FROM EMPLOYEE e JOIN JOB\_GRADES j ON e.SALARY BETWEEN j.LOWEST\_SAL AND j.HIGHEST\_SAL;

A computer screen with green text

Description automatically generated

17. Display the employees name along with their manager’s name. (use SELF JOIN)

🡺 SELECT e1.FIRST\_NAME AS Employee\_Name, e2.FIRST\_NAME AS Manager\_Name FROM EMPLOYEE e1 JOIN EMPLOYEE e2 ON e1.MANAGER\_ID = e2.EMPLOYEE\_ID;

A screen shot of a computer

Description automatically generated

18. Display emp\_id, f\_name, d\_loc, and hod\_id (using natural join).

🡺 SELECT EMPLOYEE.EMPLOYEE\_ID, EMPLOYEE.FIRST\_NAME, DEPARTMENT.D\_LOC, DEPARTMENT.HOD\_ID FROM EMPLOYEE NATURAL JOIN DEPARTMENT;

A screenshot of a computer

Description automatically generated

19. Display the employees f\_name, city and state in which they live (using natural join).

🡺 SELECT EMPLOYEE.FIRST\_NAME, EMP\_ADDRESS.CITY, EMP\_ADDRESS.STATE FROM EMPLOYEE NATURAL JOIN EMP\_ADDRESS;

A computer screen with green text

Description automatically generated

20. Display the employees emp\_id, f\_name, d\_loc, hod\_id using inner join.

🡺 SELECT EMPLOYEE.EMPLOYEE\_ID, EMPLOYEE.FIRST\_NAME, DEPARTMENT.D\_LOC, DEPARTMENT.HOD\_ID FROM EMPLOYEE INNER JOIN DEPARTMENT ON EMPLOYEE.DEPT = DEPARTMENT.D\_NAME;

A screen shot of a computer

Description automatically generated

21. Display the employees f\_name, city and state in which they live (using inner join).

🡺 SELECT EMPLOYEE.FIRST\_NAME, EMP\_ADDRESS.CITY, EMP\_ADDRESS.STATE FROM EMPLOYEE INNER JOIN EMP\_ADDRESS ON EMPLOYEE.EMP\_ID = EMP\_ADDRESS.EMP\_ID;

22. Display the employees f\_name, city and state in which they live (using join keyword).

🡺 SELECT EMPLOYEE.FIRST\_NAME, EMP\_ADDRESS.CITY, EMP\_ADDRESS.STATE FROM EMPLOYEE JOIN EMP\_ADDRESS ON EMPLOYEE.EMP\_ID = EMP\_ADDRESS.EMP\_ID;

23. Insert the following two rows in the employee table without inserting any value in the

department field.

A number of numbers and a few words

Description automatically generated with medium confidence

🡺 INSERT INTO EMPLOYEE VALUES(20,'ALEX',NULL,'ENGINEER',28000,2000,NULL,1,TO\_DATE('31-JAN-2017','DD-MM-YYYY'));

INSERT INTO EMPLOYEE VALUES(21,'PRIYA','PATEL','CLERK',12000,500,NULL,1,TO\_DATE('01-APR-2017','DD-MM-YYYY'));

A black screen with green text

Description automatically generated

24. Insert the following two rows into the department table.

A group of black text

Description automatically generated

🡺 INSERT INTO DEPARTMENT VALUES('TRAINING','MUMBAI',1);

INSERT INTO DEPARTMENT VALUES('PLACEMENT','MUMBAI',1);

A black background with green text

Description automatically generated

25. Display the employees f\_name, city and state in which they live after joining employee

and employee\_address table using left outer join.

🡺 SELECT EMPLOYEE.FIRST\_NAME, EMP\_ADDRESS.CITY, EMP\_ADDRESS.STATE FROM EMPLOYEE LEFT JOIN EMP\_ADDRESS ON EMPLOYEE.EMP\_ID = EMP\_ADDRESS.EMP\_ID;

26. Display the employees f\_name and their work location after joining employee and

department table using left join.

🡺 SELECT EMPLOYEE.FIRST\_NAME, DEPARTMENT.D\_LOC AS WORK\_LOCATION FROM EMPLOYEE LEFT JOIN DEPARTMENT ON EMPLOYEE.DEPT = DEPARTMENT.D\_NAME;

A computer screen with green text

Description automatically generated

27. Display the employees f\_name and their work location after joining employee and

department table using right join.

🡺 SELECT EMPLOYEE.FIRST\_NAME, DEPARTMENT.D\_LOC AS WORK\_LOCATION FROM EMPLOYEE RIGHT JOIN DEPARTMENT ON EMPLOYEE.DEPT = DEPARTMENT.D\_NAME;

A black screen with green text

Description automatically generated

28. Display the employees f\_name and their work location after joining employee and

department table using full join/full outer join.

🡺 SELECT EMPLOYEE.FIRST\_NAME, DEPARTMENT.D\_LOC AS WORK\_LOCATION FROM EMPLOYEE FULL JOIN DEPARTMENT ON EMPLOYEE.DEPT = DEPARTMENT.D\_NAME;

A black screen with green text

Description automatically generated

29. Find the employees who are working in their home city.

🡺 SELECT e.EMP\_ID, e.FIRST\_NAME, e.LAST\_NAME FROM EMPLOYEE e JOIN DEPARTMENT d ON e.DEPT = d.D\_NAME JOIN EMP\_ADDRESS ea ON e.EMP\_ID = ea.EMP\_ID WHERE d.D\_LOC = ea.CITY;

30. Find the job type having the minimum average salary according to jobtypes.

🡺 SELECT JOB\_TYPE, AVG(SALARY) AS AVERAGE\_SALARY FROM EMPLOYEE GROUP BY JOB\_TYPE ORDER BY AVG(SALARY) ASC FETCH FIRST 1 ROW ONLY;

# **Assignment-VI**

Name:- Bibek Chand Sah

Roll:- 22054029

Section:- CSE-05

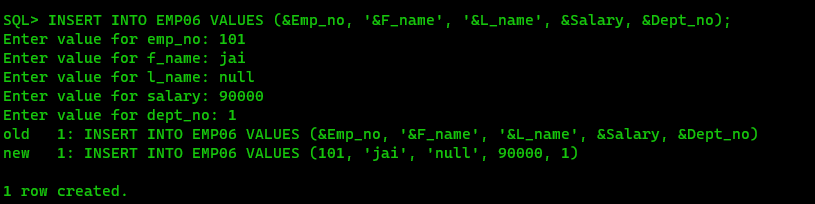
1. Create and populate the following table ‘EMP06’. Make Emp\_no the primary key and F\_name not null.

A table of numbers and letters

Description automatically generated with medium confidence

🡺 CREATE TABLE EMP06 (Emp\_no INT PRIMARY KEY, F\_name VARCHAR(50) NOT NULL, L\_name VARCHAR(50), Salary DECIMAL(10, 2), Dept\_no INT);

INSERT INTO EMP06 VALUES (&Emp\_no, '&F\_name', '&L\_name', &Salary, &Dept\_no);



2. Create and populate the following table ‘PROJECT’. Make P\_no the primary key and put a default value constraint on P\_Loc with value=‘Mumbai’.

A close-up of a number

Description automatically generated

🡺 CREATE TABLE PROJECT (P\_no INT PRIMARY KEY, P\_name VARCHAR(50), P\_Loc VARCHAR(50) DEFAULT 'Mumbai');

INSERT INTO PROJECT VALUES (&P\_no, ‘&P\_name’, ‘&P\_Loc’);

A computer screen with green text

Description automatically generated

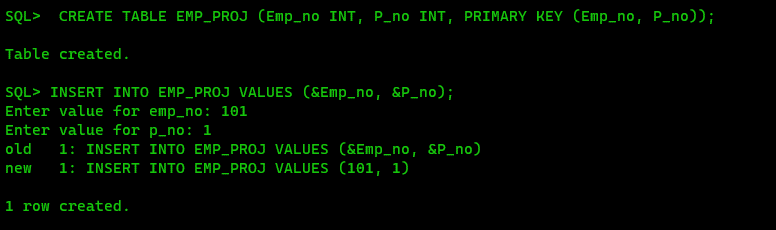
3. Create and populate the following EMP\_PROJ table. Make(Emp\_no, P\_no) the primary key.

A white sheet with black lines

Description automatically generated

🡺 CREATE TABLE EMP\_PROJ (Emp\_no INT, P\_no INT, PRIMARY KEY (Emp\_no, P\_no));

INSERT INTO EMP\_PROJ VALUES (&Emp\_no, &P\_no);



4. Display the employee’s first names with the project name’s they are working on.

🡺 SELECT e.first\_name, p.project\_name FROM EMP06 e JOIN Project\_Assignment pa ON e.employee\_id = pa.employee\_id JOIN Project p ON pa.project\_id = p.project\_id;

5. In which city Gabbar Singh works.

🡺 SELECT e.city FROM Employee e WHERE e.first\_name = 'Gabbar' AND e.last\_name = 'Singh';

6. Find the employee names who are not yet assigned to any project (using minus).

🡺 SELECT first\_name, last\_name FROM Employee MINUS SELECT e.first\_name, e.last\_name FROM Employee e JOIN Project\_Assignment pa ON e.employee\_id = pa.employee\_id;

7. Find the employee names who are not yet assigned to any project (using outer join).

🡺 SELECT e.first\_name, e.last\_name FROM Employee e LEFT JOIN Project\_Assignment pa ON e.employee\_id = pa.employee\_id WHERE pa.employee\_id IS NULL;

8. Find the project names where no employees are working (using outer join).

🡺 SELECT p.project\_name FROM Project p LEFT JOIN Project\_Assignment pa ON p.project\_id = pa.project\_id WHERE pa.project\_id IS NULL;

9. Find all the employee names who are working in project number 1 and project ‘ABC’ (using union).

🡺 (SELECT e.first\_name, e.last\_name FROM Employee e JOIN Project\_Assignment pa ON e.employee\_id = pa.employee\_id JOIN Project p ON pa.project\_id = p.project\_id WHERE p.project\_id = 1) UNION (SELECT e.first\_name, e.last\_name FROM Employee e JOIN Project\_Assignment pa ON e.employee\_id = pa.employee\_id JOIN Project p ON pa.project\_id = p.project\_id WHERE p.project\_name = 'ABC');

10. Find all the employee names who are working in both project number 1 and project number 2 (using intersect).

🡺 (SELECT e.first\_name, e.last\_name FROM Employee e JOIN Project\_Assignment pa ON e.employee\_id = pa.employee\_id WHERE pa.project\_id = 1) INTERSECT (SELECT e.first\_name, e.last\_name FROM Employee e JOIN Project\_Assignment pa ON e.employee\_id = pa.employee\_id WHERE pa.project\_id = 2);

11. Find the number of employees working in each project.

🡺 SELECT p.project\_name, COUNT(pa.employee\_id) AS num\_employees FROM Project p JOIN Project\_Assignment pa ON p.project\_id = pa.project\_id GROUP BY p.project\_name;

12. Find the average salary of each department.

🡺 SELECT d.dept\_name, AVG(e.salary) AS avg\_salary FROM Department d JOIN Employee e ON d.dept\_id = e.dept\_id GROUP BY d.dept\_name;

13. Find the department number with the number of employees working in each department where the average salary is greater than 60000 and number of employees greater than 1.

🡺 SELECT d.dept\_id, COUNT(e.employee\_id) AS num\_employees FROM Department d JOIN Employee e ON d.dept\_id = e.dept\_id GROUP BY d.dept\_id HAVING AVG(e.salary) > 60000 AND COUNT(e.employee\_id) > 1;

14. Find all the employees who earn more than Basanti.

🡺 SELECT e.first\_name, e.last\_name FROM Employee e WHERE e.salary > (SELECT salary FROM Employee WHERE first\_name = 'Basanti');

15. Find all the employees who earn more than the average salary of all employees.

🡺 SELECT e.first\_name, e.last\_name FROM Employee e WHERE e.salary > (SELECT AVG(salary) FROM Employee);

16. Find the employee who earns the highest salary.

🡺 SELECT first\_name, last\_name FROM Employee WHERE salary = (SELECT MAX(salary) FROM Employee);

17. Find the employee who earns the highest salary in dept\_no 3.

🡺 SELECT first\_name, last\_name FROM Employee WHERE salary = (SELECT MAX(salary) FROM Employee WHERE dept\_id = 3);

18. Find the employee earning the second highest salary.

🡺 SELECT first\_name, last\_name FROM Employee WHERE salary = (SELECT MAX(salary) FROM Employee WHERE salary < (SELECT MAX(salary) FROM Employee));

19. Find the dept\_no having the highest average salary.

🡺 SELECT dept\_id FROM Employee GROUP BY dept\_id HAVING AVG(salary) = (SELECT MAX(avg\_salary) FROM (SELECT AVG(salary) AS avg\_salary FROM Employee GROUP BY dept\_id));

20. Find the employee with the third highest salary among all the employees

🡺 SELECT first\_name, last\_name FROM Employee WHERE salary = (SELECT DISTINCT salary FROM Employee ORDER BY salary DESC LIMIT 1 OFFSET 2);