1. Consider the Insurance database given below.

PERSON(driver_ID, name, address)

CAR(regno, model,year)

ACCIDENT(report_number,accd_date,location)

OWNS(driver_id,regno)

PARTICIPATED(driver_id,regno,report_number,damage_amount)

i. Specify the primary keys and foreign keys and enter at least five tuples for each relation.

ii. Update the damage amount for the car with specific regno in the accident with report number 1025.

iii. Add a new accident to the database.

iv. Find the total number of people who owned cars that were involved in accidents in the year 2018.

v. Find the number of accidents in which cars belonging Wagon R were involved.

vi. Write a procedure program to find square of a number

i. Primary keys and foreign keys for each relation, along with five tuples for each relation:

1. PERSON:

Primary Key: driver_ID

Sample Tuples:

- (1, "John Smith", "123 Main St")
- (2, "Alice Johnson", "456 Elm St")
- (3, "Bob Brown", "789 Oak St")
- (4, "Mary Davis", "101 Pine St")
- (5, "David Wilson", "202 Cedar St")

2. CAR:

Primary Key: regno

Sample Tuples:

- ("ABC123", "Toyota Corolla", 2019)
- ("XYZ456", "Honda Civic", 2020)
- ("DEF789", "Ford Mustang", 2018)
- ("LMN012", "Chevrolet Malibu", 2021)
- ("PQR345", "Nissan Altima", 2019)

3. ACCIDENT:

Primary Key: report_number

Sample Tuples:

- (1025, "2023-01-15", "Intersection A")
- (2034, "2023-03-20", "Highway B")
- (3043, "2022-12-05", "Street C")
- (4052, "2021-10-10", "Parking Lot D")
- (5061, "2020-11-30", "Highway E")

4. OWNS:

Foreign Key: driver_id (References PERSON)

Foreign Key: regno (References CAR)

Sample Tuples:

- (1, "ABC123")
- (2, "XYZ456")
- (3, "DEF789")
- (4, "LMN012")
- (5, "PQR345")

5. PARTICIPATED:

Foreign Key: driver_id (References PERSON)

Foreign Key: regno (References CAR)

Foreign Key: report_number (References ACCIDENT)

Sample Tuples:

- (1, "ABC123", 1025, 500)
- (2, "XYZ456", 2034, 1000)
- (3, "DEF789", 3043, 750)
- (4, "LMN012", 4052, 1200)
- (5, "PQR345", 5061, 800)

ii. To update the damage amount for the car with specific regno in the accident with report number 1025, you can use SQL:

```sql
UPDATE PARTICIPATED
SET damage_amount = NEW_DAMAGE_AMOUNT
WHERE regno = 'ABC123' AND report_number = 1025;
```

Replace `NEW_DAMAGE_AMOUNT` with the desired new value.

iii. To add a new accident to the database, you can use SQL:

```sql
INSERT INTO ACCIDENT (report_number, accd_date, location)
VALUES (6070, '2023-05-10', 'Intersection F');
```

You would specify the appropriate values for report_number, accd_date, and location.

iv. To find the total number of people who owned cars that were involved in accidents in the year 2018, you can use SQL:

```sql
SELECT COUNT(DISTINCT P.driver_ID)
FROM PERSON P
JOIN OWNS O ON P.driver_ID = O.driver_id
JOIN ACCIDENT A ON O.regno = A.regno
WHERE YEAR(A.accd_date) = 2018;
```

v. To find the number of accidents in which cars belonging to "Wagon R" were involved, you can use SQL:

```sql
SELECT COUNT(*)
FROM CAR C
JOIN OWNS O ON C.regno = O.regno
JOIN PARTICIPATED P ON O.driver_id = P.driver_id AND O.regno = P.regno
JOIN ACCIDENT A ON P.report_number = A.report_number
WHERE C.model = 'Wagon R';
```

vi. Here's a simple Python function to find the square of a number:

```python
def find_square(number):
    return number * number

# Example usage:
```

```
result = find_square(5)
print("Square:", result)  # Output: Square: 25
```

This function takes a number as input and returns its square

---

2. Create the Book database and do the following:
Book (book_name,author_name,price,quantity).
i. Write a query to update the quantity by double in the table book.
ii. List all the book_name whose price is greater than those of book named "Database for Dummies".
iii. Retrieve the list of author_name whose first letter is 'a' along with the book_name and price.
iv. Write a function to find maximum of two numbers

To accomplish the tasks with the "Book" database, you can use SQL queries and a Python function as follows:

1. Create the "Book" table:

```sql
CREATE TABLE Book (
    book_name VARCHAR(255),
    author_name VARCHAR(255),
    price DECIMAL(10, 2),
    quantity INT
);
```

2. Write a query to update the quantity by doubling it for all books in the "Book" table:

```sql
UPDATE Book
SET quantity = quantity * 2;
```

3. List all the "book_name" whose price is greater than those of the book named "Database for Dummies":

```sql
SELECT book_name
FROM Book
WHERE price > (SELECT price FROM Book WHERE book_name = 'Database for Dummies');
```

4. Retrieve the list of "author_name" whose first letter is 'a' along with the corresponding "book_name" and "price":

```sql
SELECT author_name, book_name, price
FROM Book
WHERE author_name LIKE 'A%' OR author_name LIKE 'a%';
```

5. Write a Python function to find the maximum of two numbers:

```python
def find_maximum(a, b):
```

```
    return max(a, b)


# Example usage:
result = find_maximum(10, 25)
print("Maximum:", result)  # Output: Maximum: 25
```

This function, `find_maximum`, takes two numbers as input and returns the maximum of the two.

---

3)

Create the Company database with the following tables and do the following:
Administration(employee_salary, development _cost, fund_amount, turn_over,bonus)
Emp_details (emp_no, emp_name, DOB, address, doj, mobile_no, dept_no, salary).
i. Calculate the total and average salary amount of the employees of each department.
ii. Display total salary spent for employees.
iii. Develop a PL/SQL function to display minimum of two numbers

To create the "Company" database and perform the requested tasks, you will use SQL queries and a PL/SQL function:

1. Create the "Administration" table and "Emp_details" table:

```sql
CREATE TABLE Administration (
    employee_salary DECIMAL(10, 2),
    development_cost DECIMAL(10, 2),
    fund_amount DECIMAL(10, 2),
```

```
    turn_over DECIMAL(10, 2),
    bonus DECIMAL(10, 2)
);

CREATE TABLE Emp_details (
    emp_no INT PRIMARY KEY,
    emp_name VARCHAR(255),
    DOB DATE,
    address VARCHAR(255),
    doj DATE,
    mobile_no VARCHAR(15),
    dept_no INT,
    salary DECIMAL(10, 2)
);
```

Now, let's perform the requested tasks:

i. Calculate the total and average salary amount of the employees of each department:

```sql
SELECT dept_no, SUM(salary) AS total_salary, AVG(salary) AS average_salary
FROM Emp_details
GROUP BY dept_no;
```

ii. Display the total salary spent on all employees:

```sql
SELECT SUM(salary) AS total_salary_spent
FROM Emp_details;
```

iii. Develop a PL/SQL function to display the minimum of two numbers. PL/SQL code might look like this:

```sql
-- Create a PL/SQL function to find the minimum of two numbers
CREATE OR REPLACE FUNCTION find_minimum(a NUMBER, b NUMBER) RETURN NUMBER IS
  min_num NUMBER;
BEGIN
  IF a < b THEN
    min_num := a;
  ELSE
    min_num := b;
  END IF;
  RETURN min_num;
END find_minimum;
```

You can call this PL/SQL function as follows:

```sql
-- Example usage of the PL/SQL function to find the minimum of two numbers
```

```
DECLARE
  result NUMBER;
BEGIN
  result := find_minimum(10, 25);
  DBMS_OUTPUT.PUT_LINE('Minimum: ' || result);
END;
```

This PL/SQL function, `find_minimum`, takes two numbers as input and returns the minimum of the two. The example usage shows how to call the function and display the result.

---

4. Create the student database with the following tables and do the following:
assessment(reg_no,name, mark1, mark2, mark3, total)
dept_details (dept_no, dept_name, location).
i. Using alter command drop the column location from the table dept_details.
ii. Display all dept_name along withdept_no.
iii. Drop the table dept_details.
iv. Write a PL/SQL Trigger to update and insert a data

To work with the "student" database and perform the requested tasks, you can use SQL queries and PL/SQL statements:

1. Using the ALTER command to drop the column "location" from the "dept_details" table:

```sql
ALTER TABLE dept_details
DROP COLUMN location;
```

This command removes the "location" column from the "dept_details" table.

2. Display all "dept_name" along with "dept_no":

```sql
SELECT dept_no, dept_name
FROM dept_details;
```

This query retrieves the "dept_no" and "dept_name" columns from the "dept_details" table.

3. To drop the table "dept_details," you can use the following SQL command:

```sql
DROP TABLE dept_details;
```

This command deletes the "dept_details" table and all its data.

4. Writing a PL/SQL Trigger to update and insert data is a more complex task, and the specific requirements for this trigger are not provided. Here's an example of a simple PL/SQL trigger that updates a "total" column in the "assessment" table when a new record is inserted. This trigger assumes that you want to calculate the total as the sum of "mark1," "mark2," and "mark3":

```sql
-- Create a trigger to update the "total" column in the "assessment" table
-- when a new record is inserted.
CREATE OR REPLACE TRIGGER update_total_trigger
BEFORE INSERT ON assessment
```

```
FOR EACH ROW

BEGIN

  :NEW.total := :NEW.mark1 + :NEW.mark2 + :NEW.mark3;

END;

/
```

This trigger calculates the "total" by summing the "mark1," "mark2," and "mark3" values when a new record is inserted into the "assessment" table. You may need to adapt this example to suit your specific requirements for data updates and inserts.

---

5. Consider the following Tables for a bus reservation system application:
BUS (ROUTENO, SOURCE, DESTINATION)
PASSENGER (PID, PNAME, DOB, GENDER)
BOOK_TICKET (PID, ROUTENO, JOURNEY_DATE, SEAT_NO)
i. Include constraint that DOB of passenger should be after 2000
ii. Display the passengers who had booked the journey from Mumbai to Chennai on 02-Feb-2019
iii. List the details of passengers who have traveled more than three times on the same route.
iv. Create a View that displays the RouteNo, source, destination and journey_date which moves from Chennai to Delhi.
v. Develop a PL/SQL function to display maximum of two numbers

To address the tasks for the bus reservation system application, you can use SQL queries, constraints, and a PL/SQL function. Here's how you can achieve each task:

1. Include a constraint that DOB of a passenger should be after 2000:

```sql
ALTER TABLE PASSENGER

ADD CONSTRAINT dob_check

CHECK (DOB > DATE '2000-12-31');
```

```

This constraint ensures that the date of birth (DOB) for a passenger is after December 31, 2000.

2. Display passengers who booked the journey from Mumbai to Chennai on 02-Feb-2019:

```sql
SELECT P.PNAME, P.DOB, P.GENDER
FROM PASSENGER P
JOIN BOOK_TICKET BT ON P.PID = BT.PID
JOIN BUS B ON BT.ROUTENO = B.ROUTENO
WHERE B.SOURCE = 'Mumbai' AND B.DESTINATION = 'Chennai' AND BT.JOURNEY_DATE = DATE '2019-02-02';
```

This query retrieves passenger details who booked a journey from Mumbai to Chennai on 02-Feb-2019.

3. List the details of passengers who have traveled more than three times on the same route:

```sql
SELECT P.PID, P.PNAME, BT.ROUTENO, COUNT(*) AS travel_count
FROM PASSENGER P
JOIN BOOK_TICKET BT ON P.PID = BT.PID
GROUP BY P.PID, P.PNAME, BT.ROUTENO
HAVING COUNT(*) > 3;
```
```

This query lists passengers who have traveled more than three times on the same route.

4. Create a View that displays the RouteNo, source, destination, and journey_date for routes moving from Chennai to Delhi:

```sql
CREATE VIEW Chennai_to_Delhi_Routes AS
SELECT B.ROUTENO, B.SOURCE, B.DESTINATION, BT.JOURNEY_DATE
FROM BUS B
JOIN BOOK_TICKET BT ON B.ROUTENO = BT.ROUTENO
WHERE B.SOURCE = 'Chennai' AND B.DESTINATION = 'Delhi';
```

This view selects and displays the desired information for routes moving from Chennai to Delhi.

5. Develop a PL/SQL function to display the maximum of two numbers:

```sql
-- Create a PL/SQL function to find the maximum of two numbers
CREATE OR REPLACE FUNCTION find_maximum(a NUMBER, b NUMBER) RETURN NUMBER IS
  max_num NUMBER;
BEGIN
  IF a > b THEN
    max_num := a;
  ELSE
    max_num := b;
  END IF;
```

```
  RETURN max_num;

END find_maximum;
```

You can call this PL/SQL function to find the maximum of two numbers.

Consider the following tables.
SAILOR(sid, sname, rating, age)
BOATS(bid, bname, colour)
RESERVES(sid, bid, day)
i. List the sailors in the descending order of their rating.
ii. List the sailors whose youngest sailor for each rating and who can vote.
iii. List the sailors who have reserved for both 'RED' and 'GREEN' boats.
iv. Write a PL/SQL Trigger to update and insert a data.

To address the tasks for the given tables, you can use SQL queries for the first three tasks and create a PL/SQL trigger for the fourth task. Here's how you can achieve each task:

1. List the sailors in descending order of their rating:

```sql
SELECT sid, sname, rating, age

FROM SAILOR

ORDER BY rating DESC;
```

This query lists the sailors in descending order of their ratings.

2. List the sailors who are the youngest for each rating and are eligible to vote (assuming voting age is 18):

```sql
SELECT s1.sid, s1.sname, s1.rating, s1.age
FROM SAILOR s1
WHERE s1.age = (SELECT MIN(s2.age) FROM SAILOR s2 WHERE s2.rating = s1.rating)
AND s1.age >= 18;
```

This query selects the youngest sailor for each rating who is eligible to vote.

3. List the sailors who have reserved both 'RED' and 'GREEN' boats:

```sql
SELECT sid, sname
FROM SAILOR
WHERE sid IN (
  SELECT R1.sid
  FROM RESERVES R1
  WHERE R1.bid = (SELECT bid FROM BOATS WHERE bname = 'RED')
  INTERSECT
  SELECT R2.sid
  FROM RESERVES R2
  WHERE R2.bid = (SELECT bid FROM BOATS WHERE bname = 'GREEN')
);
```

This query selects sailors who have made reservations for both 'RED' and 'GREEN' boats.

4. Writing a PL/SQL trigger to update and insert data is a more complex task and depends on the specific requirements for data updates and inserts. Here's an example of a simple PL/SQL trigger that updates the rating of a sailor when a new record is inserted into the "SAILOR" table:

```sql
-- Create a trigger to update the rating of a sailor when a new record is inserted
CREATE OR REPLACE TRIGGER update_rating_trigger
BEFORE INSERT ON SAILOR
FOR EACH ROW
BEGIN
  IF :NEW.age < 25 THEN
    :NEW.rating := 'A';
  ELSE
    :NEW.rating := 'B';
  END IF;
END;
/
```

This trigger assigns a rating 'A' to sailors below the age of 25 and 'B' to sailors aged 25 or older when a new record is inserted. You may need to adapt this example to suit your specific requirements for data updates and inserts.

---

7. Consider the following relations for a transport management system application:
DRIVER (DCODE, DNAME, DOB, GENDER)
CITY (CCODE, CNAME)
TRUCK (TRUCKCODE, TTYPE)
i. Include the constraint as mentioned above and the gender of driver is always 'male'.
ii. Develop a SQL query to list the details of each driver and the number of trips traveled.
iii. Write a PL/SQL Trigger to update and insert a data.
iv. Count the number of drivers available

To address the tasks for the transport management system application, you can use SQL queries, constraints, and a PL/SQL trigger:

1. Include a constraint to ensure that the gender of a driver is always 'male':

```sql
ALTER TABLE DRIVER
ADD CONSTRAINT gender_check
CHECK (GENDER = 'male');
```

This constraint ensures that the gender of a driver is always 'male'.

2. Develop a SQL query to list the details of each driver and the number of trips traveled (assuming you have a "TRIPS" table related to drivers):

```sql
SELECT D.DCODE, D.DNAME, D.DOB, D.GENDER, COUNT(T.TRIPID) AS num_trips
FROM DRIVER D
LEFT JOIN TRIPS T ON D.DCODE = T.DRIVER_DCODE
GROUP BY D.DCODE, D.DNAME, D.DOB, D.GENDER;
```

This query lists the details of each driver and the number of trips they have traveled. Make sure you have a "TRIPS" table with a foreign key reference to drivers.

3. Writing a PL/SQL trigger to update and insert data is a more complex task and depends on the specific requirements for data updates and inserts. Here's an example of a simple PL/SQL trigger that updates the DOB of a driver when a new record is inserted into the "DRIVER" table:

```sql
-- Create a trigger to update the DOB of a driver when a new record is inserted
CREATE OR REPLACE TRIGGER update_dob_trigger
BEFORE INSERT ON DRIVER
FOR EACH ROW
BEGIN
  :NEW.DOB := TO_DATE('2000-01-01', 'YYYY-MM-DD');
END;
/
```

This trigger sets the DOB of a driver to '2000-01-01' when a new record is inserted. You can adapt this example to suit your specific data update and insert requirements.

4. To count the number of drivers available, you can use the following SQL query:

```sql
SELECT COUNT(*) AS num_drivers
FROM DRIVER;
```

This query counts the number of drivers in the "DRIVER" table, which represents the number of available drivers.

Consider the following relational schema for a banking database application:
CUSTOMER (CID, CNAME)
BRANCH (BCODE, BNAME)
ACCOUNT (ANO, ATYPE, BALANCE, CID, BCODE)
TRANSACTION (TID, ANO, TTYPE, TDATE, TAMOUNT)
i. Develop a SQL query to list the details of branches and the number of

accounts in each branch.
ii. Develop a SQL query to list the details of customers who have performed the most transactions today
iii. Develop a PL/SQL Function to maximum of two number .

To address the tasks for the banking database application, you can use SQL queries for the first two tasks and create a PL/SQL function for the third task:

1. Develop a SQL query to list the details of branches and the number of accounts in each branch:

```sql
SELECT B.BCODE, B.BNAME, COUNT(A.ANO) AS num_accounts
FROM BRANCH B
LEFT JOIN ACCOUNT A ON B.BCODE = A.BCODE
GROUP BY B.BCODE, B.BNAME;
```

This query lists the details of branches along with the number of accounts in each branch. It uses a LEFT JOIN to include branches even if they have no associated accounts.

2. Develop a SQL query to list the details of customers who have performed the most transactions today:

```sql
WITH DailyTransactionCounts AS (
  SELECT C.CID, C.CNAME, COUNT(*) AS num_transactions
  FROM CUSTOMER C
  JOIN ACCOUNT A ON C.CID = A.CID
  JOIN TRANSACTION T ON A.ANO = T.ANO
```

```
    WHERE TRUNC(T.TDATE) = TRUNC(SYSDATE)  -- Assuming you want today's transactions
    GROUP BY C.CID, C.CNAME
)
SELECT CID, CNAME, num_transactions
FROM DailyTransactionCounts
WHERE num_transactions = (SELECT MAX(num_transactions) FROM DailyTransactionCounts);
```

This query calculates the number of transactions performed by each customer today and selects those with the highest number of transactions.

3. Develop a PL/SQL function to find the maximum of two numbers. Here's an example of a PL/SQL function for this purpose:

```sql
-- Create a PL/SQL function to find the maximum of two numbers
CREATE OR REPLACE FUNCTION find_maximum(a NUMBER, b NUMBER) RETURN NUMBER IS
  max_num NUMBER;
BEGIN
  IF a > b THEN
    max_num := a;
  ELSE
    max_num := b;
  END IF;
  RETURN max_num;
END find_maximum;
```

This PL/SQL function takes two numbers as input and returns the maximum of the two. You can call this function to find the maximum of two numbers.

---

9)Consider the following database of student enrollment in courses and books adopted for that course.
STUDENT(regno, name, major, bdate)
COURSE(courseno, cname, dept)
ENROLL(regno, courseno, sem, marks)
i. Display the total number of students register for more than two courses in a department specified.
ii. Display the students who have secured the highest mark in each course
iii. List the youngest student of each course in all departments.
iv. Develop a PL/SQL function to display maximum of two numbers

To address the tasks for the student enrollment and courses database, you can use SQL queries for the first three tasks and create a PL/SQL function for the fourth task:

1. Display the total number of students registered for more than two courses in a specified department (e.g., 'Computer Science'):

```sql
SELECT D.dept, COUNT(E.regno) AS num_students
FROM (
  SELECT regno
  FROM ENROLL
  GROUP BY regno
  HAVING COUNT(DISTINCT courseno) > 2
) E
JOIN COURSE C ON C.courseno = E.courseno
WHERE C.dept = 'Computer Science';
```

This query counts the number of students who are registered for more than two courses in the 'Computer Science' department.

2. Display the students who have secured the highest mark in each course:

```sql
SELECT C.courseno, E.regno, E.marks
FROM ENROLL E
JOIN (
  SELECT courseno, MAX(marks) AS max_marks
  FROM ENROLL
  GROUP BY courseno
) MaxMarks ON E.courseno = MaxMarks.courseno AND E.marks = MaxMarks.max_marks
JOIN COURSE C ON C.courseno = E.courseno;
```

This query lists students who have secured the highest mark in each course.

3. List the youngest student of each course in all departments:

```sql
SELECT C.courseno, S.regno, S.name, S.bdate
FROM (
  SELECT courseno, MIN(bdate) AS min_bdate
  FROM STUDENT
  GROUP BY courseno
) YoungestStudents
JOIN STUDENT S ON S.bdate = YoungestStudents.min_bdate
JOIN COURSE C ON C.courseno = YoungestStudents.courseno;
```

```

This query lists the youngest student for each course in all departments.

4. Develop a PL/SQL function to display the maximum of two numbers:

```sql
-- Create a PL/SQL function to find the maximum of two numbers
CREATE OR REPLACE FUNCTION find_maximum(a NUMBER, b NUMBER) RETURN NUMBER IS
  max_num NUMBER;
BEGIN
  IF a > b THEN
    max_num := a;
  ELSE
    max_num := b;
  END IF;
  RETURN max_num;
END find_maximum;
```

This PL/SQL function takes two numbers as input and returns the maximum of the two. You can call this function to find the maximum of two numbers.

---

10. The following are maintained by a book dealer.
AUTHOR(author_id, name, city, country)
PUBLISHER(publisher_id, name, city, country)
CATALOG(book_id, title, author_id, publisher_id , category_id, year, price)
CATEGORY(category_id, description)
ORDER_DETAILS(order_no, book_id, quantity)
i. List the author of the book that has minimum sales.
ii. Display total number of books in each category.
```

iii. Develop a PL/SQL procedure to square of a number

To address the tasks for the book dealer's database, you can use SQL queries for the first two tasks and create a PL/SQL procedure for the third task:

1. List the author of the book that has minimum sales:

```sql
SELECT A.name AS author_name
FROM AUTHOR A
JOIN CATALOG C ON A.author_id = C.author_id
LEFT JOIN (
  SELECT book_id, SUM(quantity) AS total_sales
  FROM ORDER_DETAILS
  GROUP BY book_id
) Sales ON C.book_id = Sales.book_id
WHERE total_sales IS NULL OR total_sales = (SELECT MIN(total_sales) FROM Sales);
```

This query lists the author of the book with the minimum sales. It calculates the total sales for each book using the "ORDER_DETAILS" table.

2. Display the total number of books in each category:

```sql
SELECT CAT.category_id, CAT.description, COUNT(CAT.book_id) AS total_books
FROM CATEGORY CAT
LEFT JOIN CATALOG C ON CAT.category_id = C.category_id
GROUP BY CAT.category_id, CAT.description;
```

```

This query lists the total number of books in each category. It counts the number of books in each category using the "CATALOG" and "CATEGORY" tables.

3. Develop a PL/SQL procedure to square a number:

```sql
-- Create a PL/SQL procedure to square a number
CREATE OR REPLACE PROCEDURE square_number(p_number IN NUMBER, p_result OUT NUMBER) IS
BEGIN
  p_result := p_number * p_number;
END square_number;
```

This PL/SQL procedure takes a number as input and returns its square through an output parameter. You can call this procedure to calculate the square of a number.

---

11. Create the student database with the following tables and do the following:
mark_details(reg_no,name, mark1, mark2, mark3, total)
dept_details (dept_no, dept_name, HOD)
stud_details(reg_no,name, dob, address)
i. Using alter command to assign foreign key in mark_details.
ii. Display the address of the students who have secured the top three ranks.
iii. Develop a PL/SQL procedure to square of a number

To accomplish the tasks for the student database, you can use SQL commands for the first two tasks and create a PL/SQL procedure for the third task:

1. Using the ALTER command to assign a foreign key in the "mark_details" table, you would need to specify a foreign key relationship to link "mark_details" with "stud_details." The foreign key would typically reference the "reg_no" column. Here's an example of how you can set up the foreign key:
```

```sql
ALTER TABLE mark_details
ADD CONSTRAINT fk_stud_details_reg_no
FOREIGN KEY (reg_no)
REFERENCES stud_details (reg_no);
```

This SQL command establishes a foreign key constraint in the "mark_details" table that references the "reg_no" column in the "stud_details" table.

2. Display the address of the students who have secured the top three ranks:

```sql
SELECT s.address
FROM stud_details s
WHERE s.reg_no IN (
  SELECT m.reg_no
  FROM mark_details m
  ORDER BY m.total DESC
  FETCH FIRST 3 ROWS ONLY
);
```

This query retrieves the address of the students who have secured the top three ranks based on the total marks in the "mark_details" table.

3. Develop a PL/SQL procedure to calculate the square of a number:

```sql
CREATE OR REPLACE PROCEDURE calculate_square(p_number IN NUMBER, p_result OUT NUMBER) IS
BEGIN
   p_result := p_number * p_number;
END calculate_square;
```

This PL/SQL procedure takes a number as input and returns its square through an output parameter. You can call this procedure to calculate the square of a number.

---

12. Create a database for maintaining the cloud database
PAAS_details(server, platform, startDate, endDate, rate)
SAAS_details(server, software, startDate, endDate, rate)
DAAS_details(server, database, startDate, endDate, rate)
transaction(service, logintime, logouttime)
i. List the details of the services requested from 5th Feb to 10th Feb, 2019.
ii. Display the details of the service that are least used and most used.
iii. Perform cross join on PAAS and SAAS details.
iv. Develop a PL/SQL function to find maximum of two numbers

To work with the cloud database and perform the requested tasks, you can use SQL queries for the first two tasks and create a PL/SQL function for the fourth task. However, I'll address the third task as well:

1. List the details of the services requested from 5th Feb to 10th Feb, 2019:

```sql
SELECT *
FROM transaction
WHERE logintime >= DATE '2019-02-05' AND logouttime <= DATE '2019-02-10';
```

```

This query retrieves the details of services requested between 5th Feb and 10th Feb, 2019.

2. Display the details of the service that is least used and most used:

To find the least and most used services, you can use the following queries:

For the least used service:

```sql
SELECT service, COUNT(*) AS service_usage
FROM transaction
GROUP BY service
ORDER BY service_usage ASC
FETCH FIRST 1 ROW ONLY;
```

For the most used service:

```sql
SELECT service, COUNT(*) AS service_usage
FROM transaction
GROUP BY service
ORDER BY service_usage DESC
FETCH FIRST 1 ROW ONLY;
```
```

These two queries will retrieve the service that is least used and the one that is most used.

3. Perform a cross join on PAAS and SAAS details:

A cross join, also known as a Cartesian join, combines every row from the first table with every row from the second table. Here's how you can perform a cross join between the "PAAS_details" and "SAAS_details" tables:

```sql
SELECT *
FROM PAAS_details
CROSS JOIN SAAS_details;
```

This will give you a result set where each row from "PAAS_details" is combined with every row from "SAAS_details."

4. Develop a PL/SQL function to find the maximum of two numbers:

```sql
-- Create a PL/SQL function to find the maximum of two numbers
CREATE OR REPLACE FUNCTION find_maximum(a NUMBER, b NUMBER) RETURN NUMBER IS
  max_num NUMBER;
BEGIN
  IF a > b THEN
    max_num := a;
  ELSE
```

```
    max_num := b;
  END IF;
  RETURN max_num;
END find_maximum;
```

This PL/SQL function takes two numbers as input and returns the maximum of the two. You can call this function to find the maximum of two numbers.

---

13. Create a database for Placement and Training cell.
Stud_details(regno, name, dept, percentage)
Company(companyID,name, noOfVacancy)

Training_Details(CourseID, name, Trainer)
Placed(regno, companyID,minSal)
i. List the students who are eligible for recruitment in a particular company.
ii. Display the student who has been placed with highest salary
iii. Perform left outer join.
iv. Develop a PL/SQL procedures to maximum of two numbers

To work with the Placement and Training Cell database and perform the requested tasks, you can use SQL queries for the first two tasks, perform a left outer join for the third task, and create a PL/SQL procedure for the fourth task:

1. List the students who are eligible for recruitment in a particular company (e.g., 'ABC Company'):

```sql
SELECT S.regno, S.name, S.dept, S.percentage
FROM Stud_details S
JOIN Company C ON S.percentage >= C.minPercentage
WHERE C.name = 'ABC Company';
```

This query lists the students who are eligible for recruitment in the specified company based on their percentage.

2. Display the student who has been placed with the highest salary:

```sql
SELECT S.regno, S.name, P.minSal
FROM Stud_details S
JOIN Placed P ON S.regno = P.regno
WHERE P.minSal = (SELECT MAX(minSal) FROM Placed);
```

This query retrieves the student who has been placed with the highest salary.

3. Perform a left outer join:

A left outer join combines all rows from the left table with matching rows from the right table and includes unmatched rows from the left table. Here's how you can perform a left outer join between "Stud_details" and "Placed":

```sql
SELECT S.regno, S.name, P.minSal
FROM Stud_details S
LEFT JOIN Placed P ON S.regno = P.regno;
```

This query will give you a result set that includes all students and their placement details, with students who are not placed having NULL values in the "minSal" column.

4. Develop a PL/SQL procedure to find the maximum of two numbers:

```sql
-- Create a PL/SQL procedure to find the maximum of two numbers
CREATE OR REPLACE PROCEDURE find_maximum(p_a IN NUMBER, p_b IN NUMBER, p_result OUT NUMBER) IS
BEGIN
  IF p_a > p_b THEN
    p_result := p_a;
  ELSE
    p_result := p_b;
  END IF;
END find_maximum;
```

This PL/SQL procedure takes two numbers as input and returns the maximum of the two through an output parameter. You can call this procedure to calculate the maximum of two numbers.

---

14. Create a database for Timetable generation using the following tables:
Faculty_details(FacultyID,FacultyName, dept)
Subject_details(Subcode, subtitle, dept, year)
Subject_allocated(Subcode, year, dept,FacultyID)
Timetable(period, timefrom, timeto, Subcode,year,dept)
i. Display the timetable of individual faculty
ii. Display the timetable of each class separately
iii. Display the timetable of particular subject
Iv Develop a PL/SQL procedures to maximum of two numbers

To manage the Timetable generation database, you can use SQL queries for the first three tasks, and create a PL/SQL procedure for the fourth task:

1. Display the timetable of an individual faculty (e.g., Faculty with FacultyID '101'):

```sql
SELECT T.period, T.timefrom, T.timeto, T.Subcode
FROM Timetable T
WHERE T.FacultyID = '101';
```

This query will display the timetable for the faculty with FacultyID '101'.

2. Display the timetable of each class separately (assuming 'year' is used to represent class):

```sql
SELECT T.period, T.timefrom, T.timeto, T.Subcode, T.year
FROM Timetable T
ORDER BY T.year, T.period;
```

This query lists the timetable for each class separately by ordering the results based on 'year' and 'period'.

3. Display the timetable of a particular subject (e.g., Subject with Subcode 'CS101'):

```sql
SELECT T.period, T.timefrom, T.timeto, T.Subcode
```

```
FROM Timetable T

WHERE T.Subcode = 'CS101';
```

This query shows the timetable for the subject with Subcode 'CS101'.

4. Develop a PL/SQL procedure to find the maximum of two numbers:

```sql
-- Create a PL/SQL procedure to find the maximum of two numbers

CREATE OR REPLACE PROCEDURE find_maximum(p_a IN NUMBER, p_b IN NUMBER, p_result OUT NUMBER) IS

BEGIN

  IF p_a > p_b THEN

    p_result := p_a;

  ELSE

    p_result := p_b;

  END IF;

END find_maximum;
```

This PL/SQL procedure takes two numbers as input and returns the maximum of the two through an output parameter. You can call this procedure to calculate the maximum of two numbers.

---

15. Create a Table as workers and the details are { S.No, Name, Designation, Branch } Perform the following commands:

- Alter the table by adding a column Salary
- Alter the table by modifying the column Name
- Describe the table employee

- Copy the table employee as emp
- Truncate the table
- Delete the Second row from the table
- Drop the table

Write a PL/SQL Trigger to update and insert a data.

To perform the requested commands on the "workers" table and write a PL/SQL trigger to update and insert data, you can follow these steps:

1. Alter the table by adding a column "Salary":

```sql
ALTER TABLE workers
ADD Salary NUMBER;
```

This command adds a "Salary" column to the "workers" table.

2. Alter the table by modifying the column "Name" (for example, changing its data type):

```sql
ALTER TABLE workers
MODIFY Name VARCHAR2(50);
```

This command modifies the "Name" column to have a maximum length of 50 characters.

3. Describe the table "workers":

```sql
DESC workers;
```

This command provides a description of the structure of the "workers" table.

4. Copy the table "workers" as "emp":

```sql
CREATE TABLE emp AS SELECT * FROM workers;
```

This command creates a new table "emp" with the same structure and data as the "workers" table.

5. Truncate the table:

```sql
TRUNCATE TABLE workers;
```

This command deletes all rows from the "workers" table, but retains the table structure.

6. Delete the second row from the table:

```sql
DELETE FROM workers
```

WHERE S.No = 2;
```

This command deletes the row with "S.No" equal to 2 from the "workers" table.

7. Drop the table:

```sql
DROP TABLE workers;
```

This command permanently deletes the "workers" table and all its data.

Regarding the PL/SQL trigger, you can create a simple trigger that updates a "Salary" column when a new record is inserted into the table. Here's an example:

```sql
CREATE OR REPLACE TRIGGER update_salary_trigger
BEFORE INSERT ON workers
FOR EACH ROW
BEGIN
  -- Assign a default salary value when a new record is inserted
  :NEW.Salary := 50000;
END;
/
```
```

This trigger sets a default salary value of 50000 when a new record is inserted into the "workers" table. You can modify this trigger to suit your specific requirements for data updates and inserts.

---

16. Create the following tables
student_details {register_no, student_name, DOB, address, city}
mark_details {register_no, mark1, mark2, mark3, total }

- Display only those rows whose total ranges between 250 and 300.
- Drop the table mark_details and Delete the row whose register_no=161.
- Display all details whose names begins with 'a'.

Develop a PL/SQL procedures to maximum of two numbers.

To work with the provided tables and perform the requested actions, you can use SQL queries for the first three tasks, and I'll provide an example of a PL/SQL procedure for the fourth task:

1. Display only those rows from the "mark_details" table whose total ranges between 250 and 300:

```sql
SELECT *
FROM mark_details
WHERE total BETWEEN 250 AND 300;
```

This query retrieves rows from the "mark_details" table where the "total" falls within the range of 250 to 300.

2. Drop the table "mark_details" and delete the row whose "register_no" is 161:

To drop the table, you can use the following SQL command:

```sql
DROP TABLE mark_details;
```

```

To delete the row with "register_no" equal to 161, you can use:

```sql
DELETE FROM student_details
WHERE register_no = 161;
```

Make sure to backup your data or ensure it is no longer needed before executing these commands as they are irreversible.

3. Display all details from the "student_details" table whose names begin with 'a':

```sql
SELECT *
FROM student_details
WHERE student_name LIKE 'a%';
```

This query retrieves rows from the "student_details" table where the "student_name" starts with the letter 'a'.

4. Develop a PL/SQL procedure to find the maximum of two numbers:

```sql
-- Create a PL/SQL procedure to find the maximum of two numbers
```
```

```
CREATE OR REPLACE PROCEDURE find_maximum(p_a IN NUMBER, p_b IN NUMBER, p_result
OUT NUMBER) IS

BEGIN

  IF p_a > p_b THEN

    p_result := p_a;

  ELSE

    p_result := p_b;

  END IF;

END find_maximum;
```

This PL/SQL procedure takes two numbers as input and returns the maximum of the two through an output parameter. You can call this procedure to calculate the maximum of two numbers.

---

17. Consider the following database for a Banking Enterprise.
Branch{branch_name, branch_city, assets) ACCOUNT(accno, branch_name, balance} Depositor {customer_name, accno) CUSTOMER(customer_name, customer_street, customer_city}
Loan {loan_number, branch_name, amount}
Borrower { customer_name, loan_number)}

- Create the above tables by properly specifying the primary keys and foreign

keys and enter at least five tuples for each relation.

- Find all the customers who have at least two accounts at the main branch.
- Find all the customers who have an account at all the branches located in a specific city.
- Demonstrate how you delete all account tuples at every branch located in a specific city.

Develop a PL/SQL procedures to maximum of two numbers

First, let's define the primary keys and foreign keys for the tables and enter sample data. Afterward, I'll provide SQL queries and a PL/SQL procedure for the requested tasks.

Here's how you can create the tables with primary keys and foreign keys and enter sample data:

```sql
-- Branch Table
CREATE TABLE Branch (
    branch_name VARCHAR2(50) PRIMARY KEY,
    branch_city VARCHAR2(50),
    assets NUMBER
);

-- Account Table
CREATE TABLE Account (
    accno NUMBER PRIMARY KEY,
    branch_name VARCHAR2(50),
    balance NUMBER,
    FOREIGN KEY (branch_name) REFERENCES Branch(branch_name)
);

-- Depositor Table
CREATE TABLE Depositor (
    customer_name VARCHAR2(50),
    accno NUMBER,
    PRIMARY KEY (customer_name, accno),
    FOREIGN KEY (accno) REFERENCES Account(accno)
);
```

```
-- Customer Table
CREATE TABLE Customer (
   customer_name VARCHAR2(50) PRIMARY KEY,
   customer_street VARCHAR2(100),
   customer_city VARCHAR2(50)
);

-- Loan Table
CREATE TABLE Loan (
   loan_number NUMBER PRIMARY KEY,
   branch_name VARCHAR2(50),
   amount NUMBER,
   FOREIGN KEY (branch_name) REFERENCES Branch(branch_name)
);

-- Borrower Table
CREATE TABLE Borrower (
   customer_name VARCHAR2(50),
   loan_number NUMBER,
   PRIMARY KEY (customer_name, loan_number),
   FOREIGN KEY (loan_number) REFERENCES Loan(loan_number)
);

-- Sample Data
-- Insert sample data into the tables.
-- Please adjust the data according to your requirements.
```

```
-- Branch data
INSERT INTO Branch (branch_name, branch_city, assets)
VALUES
   ('Main Branch', 'New York', 1000000),
   ('Downtown Branch', 'New York', 800000),
   ('Uptown Branch', 'New York', 750000),
   ('Downtown Branch', 'Los Angeles', 900000),
   ('Central Branch', 'Los Angeles', 950000);

-- Account data
INSERT INTO Account (accno, branch_name, balance)
VALUES
   (1, 'Main Branch', 5000),
   (2, 'Downtown Branch', 3000),
   (3, 'Uptown Branch', 4000),
   (4, 'Main Branch', 6000),
   (5, 'Downtown Branch', 2000);

-- Depositor data
INSERT INTO Depositor (customer_name, accno)
VALUES
   ('John Doe', 1),
   ('Alice Smith', 2),
   ('Bob Johnson', 3),
   ('John Doe', 4),
   ('Alice Smith', 5);
```

```
-- Customer data
INSERT INTO Customer (customer_name, customer_street, customer_city)
VALUES
   ('John Doe', '123 Main St', 'New York'),
   ('Alice Smith', '456 Elm St', 'New York'),
   ('Bob Johnson', '789 Oak St', 'Los Angeles'),
   ('Eva White', '567 Pine St', 'New York'),
   ('Mike Brown', '345 Maple St', 'Los Angeles');

-- Loan data
INSERT INTO Loan (loan_number, branch_name, amount)
VALUES
   (101, 'Main Branch', 20000),
   (102, 'Downtown Branch', 15000),
   (103, 'Uptown Branch', 25000),
   (104, 'Downtown Branch', 18000),
   (105, 'Central Branch', 22000);

-- Borrower data
INSERT INTO Borrower (customer_name, loan_number)
VALUES
   ('John Doe', 101),
   ('Alice Smith', 102),
   ('Bob Johnson', 103),
   ('John Doe', 104),
   ('Eva White', 105);
```

Now that you have created the tables and inserted sample data, you can perform the requested tasks.

1. Find all the customers who have at least two accounts at the main branch:

```sql
SELECT D.customer_name
FROM Depositor D
WHERE D.accno IN (SELECT A.accno FROM Account A WHERE A.branch_name = 'Main Branch')
GROUP BY D.customer_name
HAVING COUNT(DISTINCT D.accno) >= 2;
```

This query retrieves customers who have at least two accounts at the 'Main Branch.'

2. Find all the customers who have an account at all the branches located in a specific city (e.g., 'New York'):

```sql
SELECT C.customer_name
FROM Customer C
WHERE NOT EXISTS (
   SELECT B.branch_name
   FROM Branch B
   WHERE B.branch_city = 'New York'
   MINUS
   SELECT A.branch_name
```

```
    FROM Account A
    WHERE A.accno IN (
        SELECT D.accno
        FROM Depositor D
        WHERE D.customer_name = C.customer_name
    )
);
```

This query retrieves customers who have an account at all branches located in 'New York.'

3. Demonstrate how you delete all account tuples at every branch located in a specific city (e.g., 'Los Angeles'):

```sql
DELETE FROM Account
WHERE branch_name IN (
    SELECT branch_name
    FROM Branch
    WHERE branch_city = 'Los Angeles'
);
```

This command deletes all account records associated with branches located in 'Los Angeles.'

4. Develop a PL/SQL procedure to find the maximum of two numbers:

```sql
-- Create a PL/SQL procedure to find the maximum of two numbers

CREATE OR REPLACE PROCEDURE find_maximum(p_a IN NUMBER, p_b IN NUMBER, p_result OUT NUMBER) IS

BEGIN

  IF p_a > p_b THEN

    p_result := p_a;

  ELSE
```

---

18. Consider the following database consisting of the following tables:
Hostel (hno, hname, type [boys/girls])
Menu (hno, day, breakfast, lunch, dinner)
Warden (wname, qual, hno)
Student (sid, sname, gender, year, hno)

- Display the total number of girls and boys hostel in the college.
- Display the menu in the hostel ‘x’ on Tuesday.
- Display the number of wardens for each hostel.II.

Write a PL/SQL Trigger to update and insert a data.

To work with the provided tables and perform the requested tasks, you can use SQL queries for the first two tasks and write a PL/SQL trigger for the third task:

1. Display the total number of girls and boys hostels in the college:

```sql
SELECT type, COUNT(*) AS count

FROM Hostel

GROUP BY type;
```

This query will display the count of hostels for both boys and girls.

2. Display the menu in a specific hostel (e.g., 'x') on a specific day (e.g., 'Tuesday'):

```sql
SELECT M.*
FROM Menu M
JOIN Hostel H ON M.hno = H.hno
WHERE H.hname = 'x' AND M.day = 'Tuesday';
```

This query retrieves the menu for the 'x' hostel on Tuesday.

3. Display the number of wardens for each hostel:

```sql
SELECT H.hname, COUNT(W.wname) AS warden_count
FROM Hostel H
LEFT JOIN Warden W ON H.hno = W.hno
GROUP BY H.hname;
```

This query lists the hostels and the number of wardens for each hostel.

4. Write a PL/SQL trigger to update and insert data:

Here's an example of a PL/SQL trigger that automatically updates the "year" in the "Student" table when a new record is inserted, based on the "gender":

```sql
-- Create a PL/SQL trigger to update and insert data
CREATE OR REPLACE TRIGGER update_year_trigger
BEFORE INSERT ON Student
FOR EACH ROW
BEGIN
  IF :NEW.gender = 'boys' THEN
    :NEW.year := 'First Year'; -- Set year for boys
  ELSIF :NEW.gender = 'girls' THEN
    :NEW.year := 'Second Year'; -- Set year for girls
  END IF;
END;
/
```

This trigger updates the "year" column based on the "gender" when a new student record is inserted. You can customize this trigger to suit your specific requirements for data updates and inserts.

---

19. Create the Company database with the following tables and do the following:
Administration(employee_salary, development _cost, fund_amount, turn_over,bonus)
Emp_details (emp_no, emp_name, DOB, address, doj, mobile_no, dept_no, salary).
i. Calculate the total and average salary amount of the employees of each department.
2. Write a PHP program to create user login page

To calculate the total and average salary amount of employees in each department from the "Emp_details" table, you can use SQL queries. Here's how you can do it:

```sql
-- Calculate total and average salary amount of employees for each department
```

```
SELECT dept_no, SUM(salary) AS total_salary, AVG(salary) AS average_salary
FROM Emp_details
GROUP BY dept_no;
```

This query calculates the total and average salary for each department by grouping the records by "dept_no."

For creating a user login page in PHP, you can follow a basic example. Below is a simple PHP program to create a user login page:

```php
<!DOCTYPE html>
<html>
<head>
  <title>User Login</title>
</head>
<body>
  <?php
  // Check if the form is submitted
  if (isset($_POST['submit'])) {
    $username = $_POST['username'];
    $password = $_POST['password'];

    // Replace with your actual username and password validation logic
    if ($username === 'your_username' && $password === 'your_password') {
      echo "Login successful. Welcome, $username!";
    } else {
```

```
        echo "Login failed. Please check your credentials.";
    }
}
?>

<h2>User Login</h2>
<form method="POST" action="login.php"> <!-- Replace "login.php" with the actual filename of your PHP script -->
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" required><br><br>

    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required><br><br>

    <input type="submit" name="submit" value="Login">
</form>
</body>
</html>
```

Please replace 'your_username' and 'your_password' with actual username and password validation logic. This is a simple login form that checks the provided username and password, and it displays a message based on whether the login was successful or not. You should also replace "login.php" with the filename of your PHP script that handles the login logic.

---

20 Consider the following tables.
SAILOR(sid, sname, rating, age)
BOATS(bid, bname, colour)
RESERVES(sid, bid, day)
i. List the sailors in the descending order of their rating.
2. Write a PHP program for student information system

To list the sailors in descending order of their rating from the "SAILOR" table, you can use SQL as follows:

```sql
SELECT sid, sname, rating, age
FROM SAILOR
ORDER BY rating DESC;
```

This query retrieves sailors' data sorted by their rating in descending order.

Regarding creating a PHP program for a student information system, it's a broad and complex task that typically involves building a web application. Below is a simple example of a PHP program that allows you to enter student information (name and age) and displays it:

```php
<!DOCTYPE html>
<html>
<head>
  <title>Student Information System</title>
</head>
<body>
  <h2>Student Information System</h2>

  <form method="post" action="process.php">
    <label for="name">Student Name:</label>
    <input type="text" name="name" id="name" required><br><br>
```

```
    <label for="age">Student Age:</label>
    <input type="text" name="age" id="age" required><br><br>

    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

In the above HTML form, students can enter their name and age. The form data is sent to a PHP script named "process.php" for processing. Below is an example of how you can process and display the student information in "process.php":

```php
<!DOCTYPE html>
<html>
<head>
  <title>Student Information</title>
</head>
<body>
  <h2>Student Information</h2>

  <?php
  if (isset($_POST['name']) && isset($_POST['age'])) {
    $name = $_POST['name'];
    $age = $_POST['age'];

    echo "Student Name: $name<br>";
```

```
    echo "Student Age: $age<br>";
  }
  ?>
</body>
</html>
```

This "process.php" script retrieves the student's name and age from the form submission and displays the information. Keep in mind that this is a very basic example. In a real-world student information system, you'd typically store student data in a database, handle user authentication, and have more extensive features for data management and reporting.