B: SQL

1) What does *UNION*? What is the difference between *UNION* and *UNION ALL*? What does *INTERSECT* ? What are the prerequisites to use these statements?

-Union is the operator of the union of two or more sets (samples), the result of which is unique rows in sorted format.

-The difference between Union and Union All operators is that Union All displays all the sample values as a result.

-Intersect is a set intersection operator, the result of which are unique sorted lines.

Example: Consider we have two mobile clients(Client_1 and Client_2) with some mobile packages and we would like to get different results such as: Similar packages, Different packages, Client_1's packages, Clients_2's packages.

2) Explain shortly the idea of transaction blocks.

- The set of changes on the Database (Update, Insert or Delete).

-The executing of the transaction are operators (Commit, Rollback)

3)What does the statement *CREATE TRIGGER* ? Explain shortly.

-Before or after handlers over «Insert», «Update» or «Delete» or over database system events, for instance, «Log on» or «Log off»

Example: What does who do in Database(you can monitor this actions)

4)Write a statement to extract number of full months (result shown as integer) between two timestamps: *date1* and *date2*.

«select trunc(months_between(to_timestamp(sysdate+70),(to_timestamp(sysdate)))) from dual»

5) Explain the results from the query below. What is the name of the function in the last column? What is the difference between this function and a regular aggregate function. How else can we use this function? Give some examples.

——————————

```
    SELECT depname, empno, salary, rank() OVER (PARTITION BY depname ORDER BY salary

    DESC) FROM empsalary;
```

——————————

-It will displays a table with the columns depname, empno, salary, "rank". The result of the analytical function will be the numbers in order, built by groups from the depname field and sorted by the salary field in descending order.

-RANK - it’s an analytic function.

-The main difference between these two functions is that aggregate functions are used to return a single result value based on a group of rows. When we use the analytic function, we must use “OVER” and we also need to use order by or partition by as analytic

-As an aggregate function, "RANK" calculates the rank of a hypothetical row identified by the arguments of the function with respect to a given sort specification. The arguments of the function must all evaluate to constant expressions within each aggregate group, because they identify a single row within each group

6) Write a statement to create a multicolumn index on columns: loan_id, bank_id in loan_request table.

```
create table loan_request (loan_id number , bank_id number);

create index i_loan_request_comp on loan_request(loan_id, bank_id);
```

7) Write a statement to find all records from table person for which person_id is NOT unique.

```
select *
  from person p
 where p.preson_id in (select p.preson_id
                       from person p
                       group by preson_id
                       having count(preson_id) > 1)
```

8) Can you write a simple SQL statement which selects the last_name and application_date of customers who have applied after January 12th 2012, whereas:

-Column last_name is in table person

-Column application_date is in table account

-The connection between both tables is via the column account_id which exists in both tables

```
select p.last_name, a.application_date
  from person p, account a
 where p.account_id = a.account_id
 and  to_date(to_char(a.application_date, 'mm.dd.yyyy'), 'mm.dd.yyyy') >
	to_date('01.12.2012', 'mm.dd.yyyy')
```

9) There are given two tables (details below). Write ONE query to show for each customer: the last successful loan_request (loan_id, creation_date, loan_amount), sum of loan_amount of all failed loan_requests, ratio of successful_to_failed applications. (nice to have)

```
---the last successful loan_request (loan_id, creation_date, loan_amount)
select ca.account_id,
       lr1.loan_id,
       ext1.creation_date,
       lr1.loan_amount
  from loan_request lr1,
       (select max(lr.creation_date) creation_date, lr.account_id
          from loan_request lr
         group by lr.account_id
         where lr.loan_amount = 'SUCCESS') ext1,
       customer_account ca
 where ext1.creation_date = lr1.creation_date
   and ext.account_id = lr1.account_id
   and ca.account_id = lr1.account_id
        and ext.account_id = ca.account_id
```

__**C:R**__

1) Can you write a R statement which gives you the average, median and variance of the variable main_income

-mean(main_icome)

-median(main_income)

-sd(main_income)

-var(main_income)

OR

-summary(main_income)

2) Please interpret the following statements

a. ba_level_postbank <- ba_level_postbank[,!(names(ba_level_postbank) == "signup_date")]

-Create ba_level_postbank variable which will consist of all the rows from names column with not «signup_value» value.

b. ba_level_postbank$ec_property_owned_square_metres[is.na(ba_level_postbank$ec_proper ty_owned_square_metres)] <- 0

-From the dataset ba_level_postbank choose column ec_property_owned_square_metres. And in this column values = NaN change into zeroes (0).

c. ba_level_postbank <- subset(ba_level_postbank, received_palimony >= 0)

-Create ba_level_postbank variable which will consist of subset from all the dataset where in the column received_polimony values will be >= 0

d. ba_level_postbank$lead_canal <- as.factor(ba_level_postbank$lead_canal)

-Change the type of lead_canal column from ba_level_postbank dataset into format column

e. ba_level_postbank$effective_interest <- as.factor(ifelse(ba_level_postbank$ effective_interest <= 6, 1, ifelse(ba_level_postbank$ effective_interest <= 8, 2, 3)))

-**<u>Shortly:</u>**

«if» value in effective_interest <= 6 then 1 «elif» value in effective_interest <= 8 then 2 «else» 3

<u>-Full answer:</u>

Create a variable ba_level_postbank with such values as IF in the column effective_interest value <=6 (True) then value ‘1’ ELSE IF in the column effective_interest value <=8 (True) then value ‘2’ ELSE value ‘3’