Question paper and Answer Key

CMR
INSTITUTE OF
TECHNOLOGY

USN	1	C								

Internal Assessment Test 2 – May 2017

Sub:	**Database Management System**							**Code:**	16MCA23
Date:	09-05-17	Duration:	90 mins	Max Marks:	50	**Sem:**	II	**Branch:**	**MCA**

Note: Answer any 5 questions. All questions carry equal marks. Total marks: 50

			Marks	OBE CO	OBE RBT
1.	a.	Explain the following relational algebra operations: 1. JOIN 2.SELECT 3.Cartesian product 4. Union 5. Set difference	[10]	CO2	L3
2.	a.	Explain the characteristics of relations	[5]	CO1	L2
	b.	Discuss relational model constraints	[5]	CO1	L2
3.	a.	Explain division operator. When is this operation useful?	[5]	CO2	L4
	b.	Discuss Insertion, Deletion and updation anomalies by taking suitable examples.	[5]	CO2	L2
4.	a.	Define functional dependency. Write all its inference rules.	[10]	CO4	L2
5.	a.	What are normal forms? Explain the normal forms proposed by Codd	[10]	CO4	L3
6	a.	Define trigger? Explain with example**.**	[5]	CO4	L1
	b.	Define stored procedure? Explain with example.	[5]	CO2	L1
7	a.	Explain with example 1NF,2NF and 3 NF	[10]	CO4	L3
8	a.	Consider the following relations for a database that keeps track of business trips of sales persons in a sales office and give the corresponding relation algebra query. SALES PERSON(salespersonID, Name, Start-year, Dept-no) TRIP(salespersonID, From, To, Departure-date, Return-date, TripID) EXPENSE(TripID, Account-No, Amount) 1. Give the details (all attributes of TRIP relation) for trip that exceeded Rs.8000/- in expenses. 2. Print the 'salespersonID' and 'Name' of the salesman who took trips to "Delhi". 3. Print the total trip expenses incurred by the salesman with salesperson ID='502'	[10]	CO2	L4

Question paper and Answer Key

INSTITUTE OF TECHNOLOGY

Internal Assessment Test 2 – May 2017

Sub:	Database Management System	Code:	16MCA23

Note: Answer any 5 questions. All questions carry equal marks. Total marks: 50

1. a. **Explain the following relational algebra operations:**
1. JOIN 2.SELECT 3.Cartesian product 4. Union 5. Set difference

1. Join (2 M)

   A SQL join clause combines records from two or more tables in a relational database. It creates a set that can be saved as a table or used as it is. A JOIN is a means for combining fields from two tables (or more) by using values common to each.

2. Select(2 M)

   SELECT is used to obtain a subset of the tuples of a relation that satisfy a select condition.
   For example, find all employees born after 1st Jan 1950:
   SELECTdob '01/JAN/1950'(employee)

3. Cartesian Product(2 M)

   A Cartesian product of n sets, also known as a n-foldCartesian product, can be represented by an array of n dimensions, where each element is an n-tuple. An ordered pair is a 2-tuple or couple

4. Union(2 M)

   It performs binary union between two given relations and is defined as –

$r \cup s = \{ t \mid t \in r \text{ or } t \in s\}$

Notion – r U s

Where r and s are either database relations or relation result set (temporary relation).

For a union operation to be valid, the following conditions must hold –

- r, and s must have the same number of attributes.
- Attribute domains must be compatible.
- Duplicate tuples are automatically eliminated.

$\prod$ author (Books) $\cup$ $\prod$ author (Articles)

5. Set Difference (2 M)

   The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

   Notation – r – s

   Finds all the tuples that are present in r but not in s.

   ```
   ∏ author (Books) − ∏ author (Articles)
   ```

2. a. **Explain the characteristics of relations. (5 Marks)**

   Ordering of tuples in a relation
   Ordering of values within a tuple and an alternate definition of a relation.
   Values and Nulls in the tuple

   b. **Discuss relational model constraints. (5 Marks)**

   Inherent or Implicit Constraints
   Schema based/ explicit constraints
   Application bases/ Semantic based/business rules

3. a. **Explain division operator. When is this operation useful? (5 Marks)**

The formal definition of division is as follows:

$$A/B = \pi x(A) - \pi x((\pi x(A) \times B) - A)$$

The division operator denoted by ÷ is suited to queries that include the phrase "for all". suppose that we wish to find all customers who hav3e an account at all the branches located at Brooklyn. We obtain all branches in Booklyn by the expression

r1= ∏ br_name amount (σ br_city = "Brooklyn" (branch)

we can find all customer-name, br_name pairs for which the customer has an account at a branch by writing

r2=∏ customer_name,br_name (account |x| depositor)

now we have to find customers who appear in r2 with every branch name in r1.the operation that provide this is r2 ÷ r1.

Formally, let r(R) and s(S) be relations, and let S ⊆ R ; that is, every attribute of schema S is also in schema R. the relation r ÷ s is a relation of schema R – S (that is, on the schema containing all attributes of schema R that are not in schema S). A tuple t is in r ÷ s if and only if both of two conditions hold:

1. t is in ∏ R-S(r)
2. every tuple ts in s, there is a tuple tr in r satisfying both of the following

a. tr[S] = ts[S]

b. tr[R-S] = t

using fundamental operations we can write the division as

r ÷ s = ∏ R-S (r) - ∏ R-S ((∏ R-S (r) x s) - ∏ R-S,S (r))

The division operator is used for queries which involve the ‘all’ qualifier such as “Find the names of sailors who have reserved all boats”.

b. **Discuss Insertion, Deletion and updation anomalies by taking suitable examples**.

Insertion

It is a failure to place information about a new database entry into all the places in the database where information about the new entry needs to be stored. In a properly normalized database, information about a new entry needs to be inserted into only one place in the database, in an inadequatly normalized database, information about a new entry may need to be inserted into more than one place, and human fallibility being what it is, some of the needed additional insertions may be missed.

Deletion

It is a failure to remove information about an existing database entry when it is time to remove that entry. In a properly normalized database, information about an old, to-be-gotten-rid-of entry needs to be deleted from only one place in the database, in an inadequately normalized database, information about that old entry may need to be deleted from more than one place.

Update

An update of a database involves modifications that may be additions, deletions, or both. Thus “update anomalies” can be either of the kinds discussed above.

All three kinds of anomalies are highly undesirable, since their occurrence constitutes corruption of the database. Properly normalized database are much less susceptible to corruption than are un-normalized databases.

4. a. **Define functional dependency. Write all its inference rules**. **5 Marks**

In relational database theory, a functional dependency is a constraint between two sets of attributes in a relation from a database. In other words, functional dependency is a constraint that describes the relationship between attributes in a relation.

Let A, B and C and D be arbitrary subsets of the set of attributes of the giver relation R, and let AB be the union of A and B. Then,⇒→

Reflexivity

If B is subset of A, then A → B

Augmentation
If A → B, then AC → BC

Transitivity:
If A → B and B → C, then A → C.

Projectivity or Decomposition Rule
If A → BC, Then A → B and A → C

```
Proof  :

Step 1 : A → BC (GIVEN)

Step 2 : BC → B (Using Rule 1, since B ⊆ BC)

Step 3 : A → B (Using Rule 3, on step 1 and step 2)
```

Union or Additive Rule :
If A→B, and A→C Then A→BC.

```
Proof :

Step 1 : A → B (GIVEN)

Step 2 : A → C (given)

Step 3 : A → AB (using Rule 2 on step 1, since AA=A)

Step 4 : AB → BC (using rule 2 on step 2)

Step 5 : A → BC (using rule 3 on step 3 and step 4)
```

Pseudo Transitive Rule :
If A → B, DB → C, then DA → C

```
Proof :

Step 1 : A → B (Given)

Step 2 : DB → C (Given)

Step 3 : DA → DB (Rule 2 on step 1)

Step 4 : DA → C (Rule 3 on step 3 and step 2)
```

5. a. What are normal forms? Explain the normal forms proposed by Codd

10 Marks

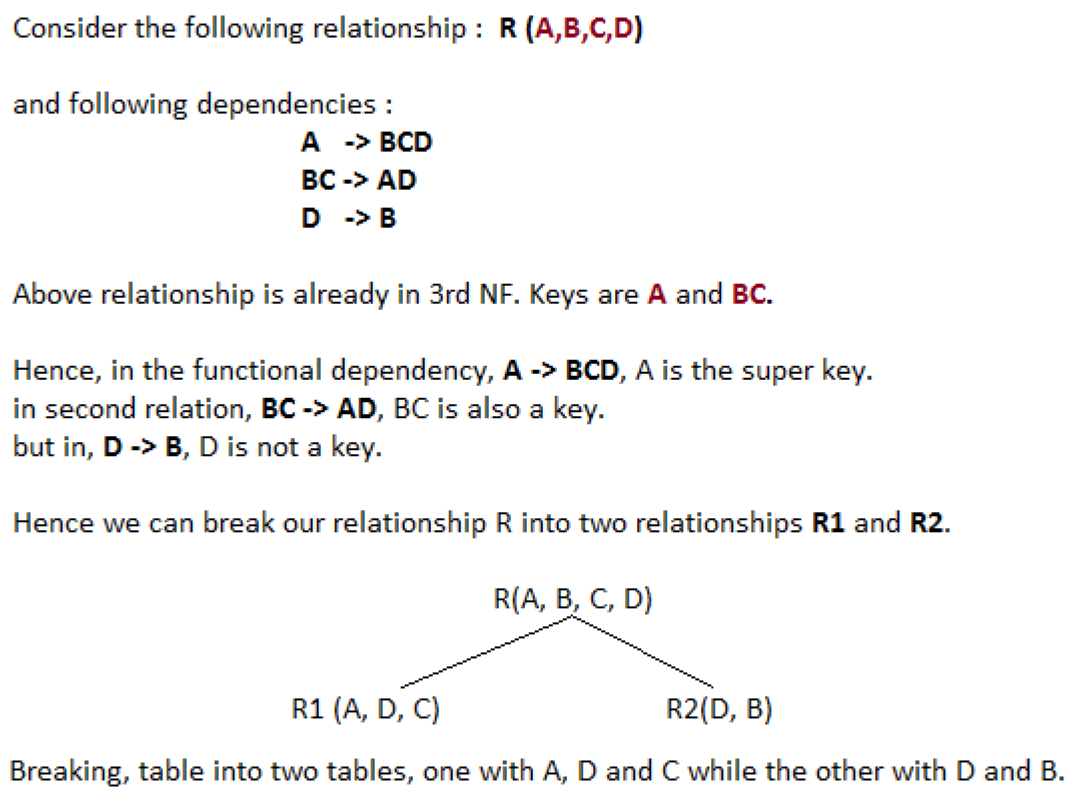
A basic objective of the first normal form defined by Codd in 1970 was to permit data to be queried and manipulated using a "universal data sub-language" grounded in first-order logic. The objectives of normalization beyond 1NF (First Normal Form) were stated as follows by Codd:

1. To free the collection of relations from undesirable insertion, update and deletion dependencies;

2. To reduce the need for restructuring the collection of relations, as new types of data are introduced, and thus increase the life span of application programs;
3. To make the relational model more informative to users;
4. To make the collection of relations neutral to the query statistics, where these statistics are liable to change as time goes by.

Boyce and Codd Normal Form is a higher version of the Third Normal form. This form deals with certain type of anamoly that is not handled by 3NF. A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF. For a table to be in BCNF, following conditions must be satisfied:

- R must be in 3rd Normal Form
- and, for each functional dependency (X -> Y), X should be a super Key.



Consider the following relationship : **R (A,B,C,D)**

and following dependencies :

A -> BCD
BC -> AD
D -> B

Above relationship is already in 3rd NF. Keys are **A** and **BC.**

Hence, in the functional dependency, **A -> BCD**, A is the super key.
in second relation, **BC -> AD**, BC is also a key.
but in, **D -> B**, D is not a key.

Hence we can break our relationship R into two relationships **R1** and **R2.**

R(A, B, C, D)

R1 (A, D, C) R2(D, B)

Breaking, table into two tables, one with A, D and C while the other with D and B.

6a **Define trigger? Explain with example.** **5 Marks**

A trigger is a special kind of stored procedure that automatically executes when an event occurs in the database server. DML triggers execute when a user tries to modify data through a data manipulation language (DML) event. DML events are INSERT, UPDATE, or DELETE statements on a table or view.

AFTER trigger

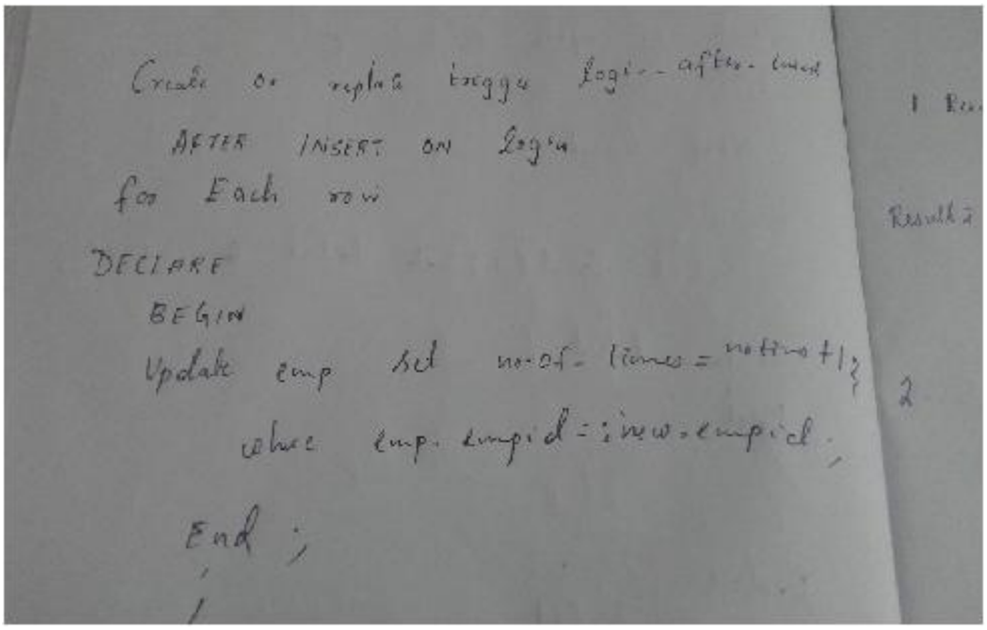
AFTER triggers are executed after the action of the INSERT, UPDATE, MERGE, or DELETE statement is performed. AFTER triggers are never executed if a constraint violation occurs; therefore, these triggers cannot be used for any processing that might prevent constraint violations. For every INSERT, UPDATE, or DELETE action specified in a MERGE statement, the corresponding trigger is fired for each DML operation.

INSTEAD OF trigger

INSTEAD OF triggers override the standard actions of the triggering statement. Therefore, they can be used to perform error or value checking on one or more columns and the perform additional actions before insert, updating or deleting the row or rows. For example, when the value being updated in an hourly wage column in a payroll table exceeds a specified value, a trigger can be defined to either produce an error message and roll back the transaction, or insert a new record into an audit trail before inserting the record into the payroll table. The primary advantage of INSTEAD OF triggers is that they enable views that would not be updatable to support updates. For example, a view based on multiple base tables must use an INSTEAD OF trigger to support inserts, updates, and deletes that reference data in more than one table. Another advantage of INSTEAD OF triggers is that they enable you to code logic that can reject parts of a batch while letting other parts of a batch to succeed.

CLR Triggers

A CLR Trigger can be either an AFTER or INSTEAD OF trigger. A CLR trigger can also be a DDL trigger. Instead of executing a Transact-SQL stored procedure, a CLR trigger executes one or more methods written in managed code that are members of an assembly created in the .NET Framework and uploaded in SQL Server.



```
Create or replace trigger login_after_insert
  AFTER INSERT ON login
for Each row
DECLARE
  BEGIN
  Update emp set no-of-times = nottimes+1;
     where emp.empid=:new.empid;
  End;
  /
```

1 Re

Result 2

2

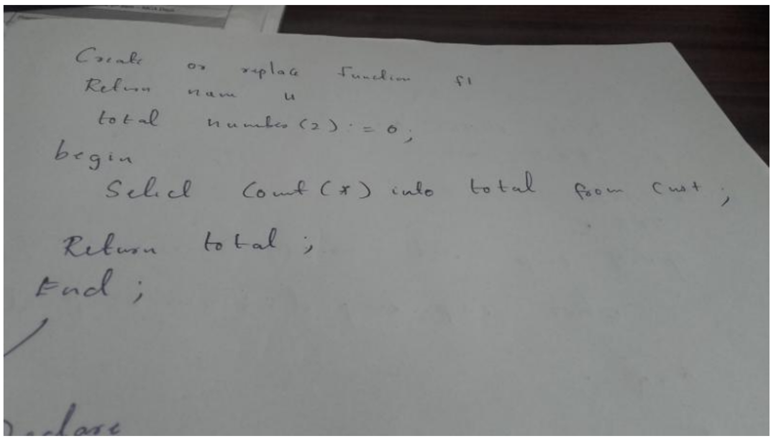
b **What is a stored procedure? Explain with example.** **5 Marks**

```
CREATE [OR REPLACE] PROCEDURE procedure_name
    [ (parameter [,parameter]) ]
IS
    [declaration_section]
BEGIN
    executable_section
[EXCEPTION
    exception_section]
END [procedure_name];
```

When we create a procedure or function, you may define parameters. There are three types of parameters that can be declared:

1. IN - The parameter can be referenced by the procedure or function. The value of the parameter can not be overwritten by the procedure or function.
2. OUT - The parameter can not be referenced by the procedure or function, but the value of the parameter can be overwritten by the procedure or function.

IN OUT - The parameter can be referenced by the procedure or function and the value of the parameter can be overwritten by the procedure or function



```
Create or replace function f1
Return num is
 total number(2) := 0;
begin
  Select count(*) into total from cust;
 Return total;
End;
/

Declare
```

7a Explain with example 1NF,2NF and 3 NF. **10 Marks**

First Normal Form (1NF)

As per First Normal Form, no two Rows of data must contain repeating group of information i.e each set of column must have a unique value, such that multiple columns cannot be used to fetch the same row. Each table should be organized into rows, and each row should have a primary key that distinguishes it as unique.

The Primary key is usually a single column, but sometimes more than one column can be combined to create a single primary key. For example consider a table which is not in First normal form

Student Table :

Student	Age	Subject
Adam	15	Biology, Maths
Alex	14	Maths
Stuart	17	Maths

In First Normal Form, any row must not have a column in which more than one value is saved, like separated with commas. Rather than that, we must separate such data into multiple rows.

Student Table following 1NF will be :

Student	Age	Subject

Adam	15	Biology
Adam	15	Maths
Alex	14	Maths
Stuart	17	Maths

Using the First Normal Form, data redundancy increases, as there will be many columns with same data in multiple rows but each row as a whole will be unique.

Second Normal Form (2NF)

As per the Second Normal Form there must not be any partial dependency of any column on primary key. It means that for a table that has concatenated primary key, each column in the table that is not part of the primary key must depend upon the entire concatenated key for its existence. If any column depends only on one part of the concatenated key, then the table fails Second normal form.

In example of First Normal Form there are two rows for Adam, to include multiple subjects that he has opted for. While this is searchable, and follows First normal form, it is an inefficient use of space. Also in the above Table in First Normal Form, while the candidate key is {Student, Subject}, Age of Student only depends on Student column, which is incorrect as per Second Normal Form. To achieve second normal form, it would be helpful to split out the subjects into an independent table, and match them up using the student names as foreign keys.

New Student Table following 2NF will be :

Student	Age
Adam	15
Alex	14
Stuart	17

In Student Table the candidate key will be Student column, because all other column i.e Age is dependent on it.

New Subject Table introduced for 2NF will be :

Student	Subject

Adam	Biology
Adam	Maths
Alex	Maths
Stuart	Maths

In Subject Table the candidate key will be {Student, Subject} column. Now, both the above tables qualifies for Second Normal Form and will never suffer from Update Anomalies. Although there are a few complex cases in which table in Second Normal Form suffers Update Anomalies, and to handle those scenarios Third Normal Form is there.

Third Normal Form (3NF)

Third Normal form applies that every non-prime attribute of table must be dependent on primary key, or we can say that, there should not be the case that a non-prime attribute is determined by another non-prime attribute. So this transitive functional dependency should be removed from the table and also the table must be in Second Normal form. For example, consider a table with following fields.

Student_Detail Table :

Student_id	Student_name	DOB	Street	city	State	Zip

In this table Student_id is Primary key, but street, city and state depends upon Zip. The dependency between zip and other fields is called transitive dependency. Hence to apply 3NF, we need to move the street, city and state to new table, with Zip as primary key.

New Student_Detail Table :

Student_id	Student_name	DOB	Zip

Address Table :

Zip	Street	city	state

The advantage of removing transtive dependency is,

- Amount of data duplication is reduced.
- Data integrity achieved.

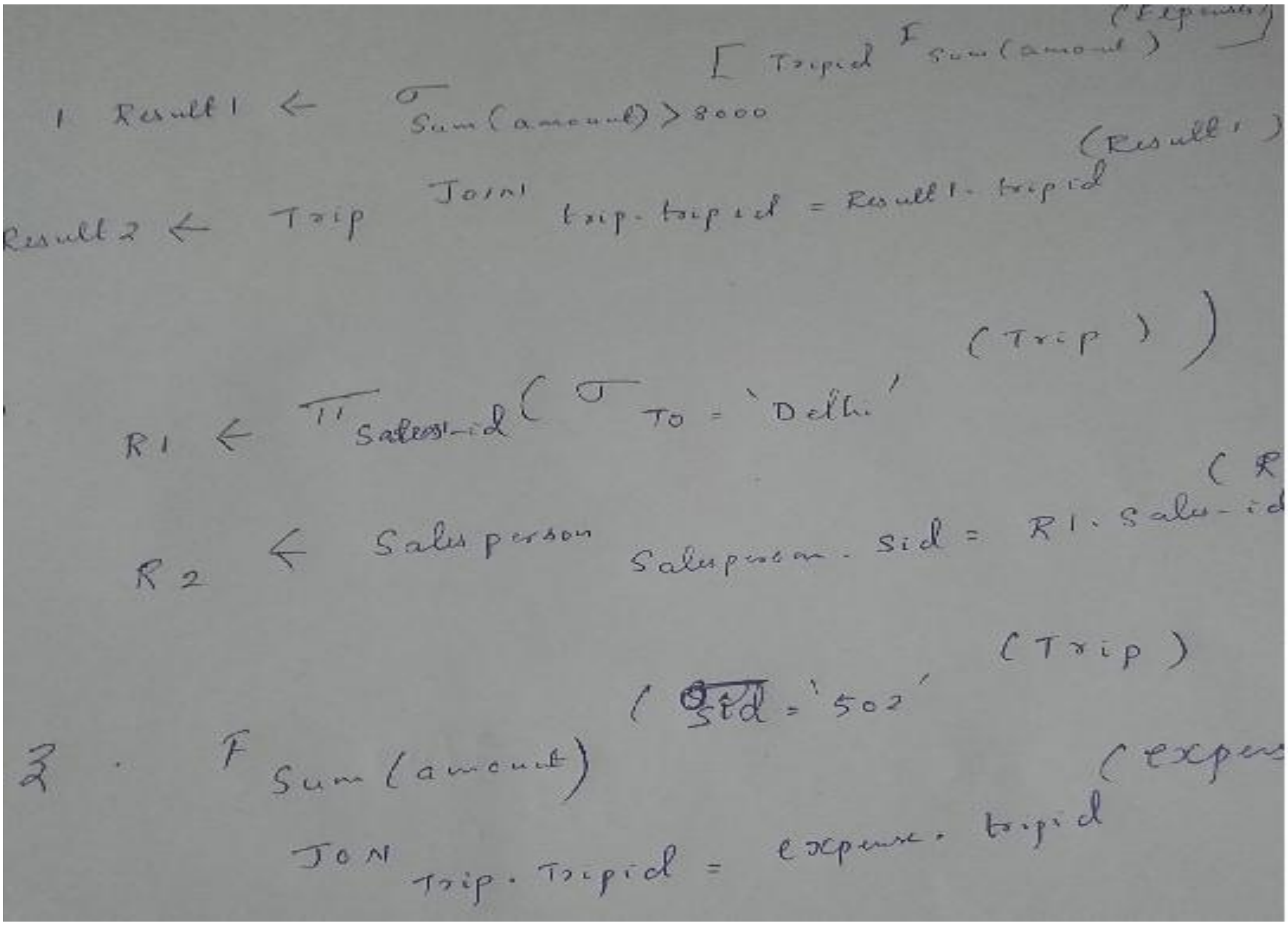
8a **Consider the following relations for a database that keeps track of business trips of sales persons in a sales office and give the corresponding relation algebra query.**

SALES PERSON(salespersonID, Name, Start-year, Dept-no)

TRIP(salespersonID, From, To, Departure-date, Return-date, TripID)

EXPENSE(TripID, Account-No, Amount)

1. Give the details (all attributes of TRIP relation) for trip that exceeded Rs.8000/- in expenses.
2. Print the 'salespersonID' and 'Name' of the salesman who took trips to "Delhi".
3. Print the total trip expenses incurred by the salesman with salesperson ID='502'.



(Expense)

[$_{\text{Tripid}}\mathcal{F}_{\text{sum(amount)}}$

1 Result1 ← $\sigma_{\text{Sum(amount)} > 8000}$

(Result1)

Result2 ← Trip Join $_{\text{trip.tripid} = \text{Result1.tripid}}$

(Trip))

R1 ← $\pi_{\text{salesp-id}}$ ($\sigma_{\text{To} = \text{'Delhi'}}$

(R

R2 ← Salesperson $_{\text{Salesperson.sid} = \text{R1.sales-id}}$

(Trip)

($\sigma_{\text{sid} = \text{'502'}}$

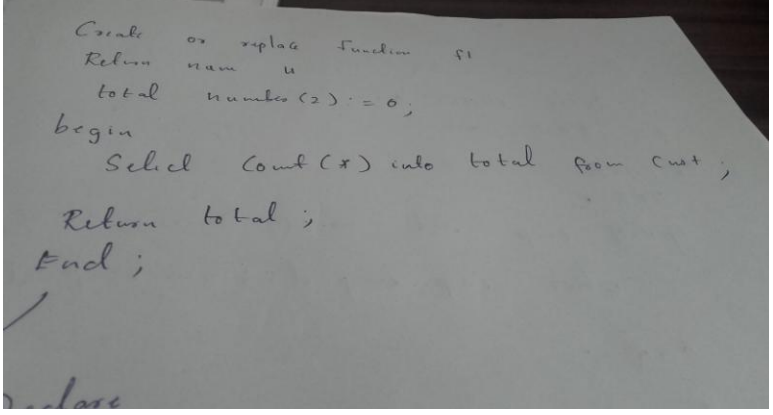
3 . $\mathcal{F}_{\text{Sum(amount)}}$

(expens

JOIN $_{\text{Trip.Tripid} = \text{expense.tripid}}$

Question paper and Answer Key

10 Marks



```
Create or replace function f1
 Return num is
  total number(2) := 0;
begin
   Select Count(*) into total from cust;
 Return total;
End;
```

Declare