

MINI PROJECT

IN

DATABASE MANAGEMENT

SYSTEMS

COURIER MANAGEMENT

SYSTEM

# INDEX

S .no	Title	Page.no
1.	Abstract	3
2.	Requirements	4
3.	Entities of the database	5
4.	E-R Diagrams of the database	7
5.	Total E-R diagram of the database	9
6.	Schema of the database	10
7.	Creating Tables	11
8.	Applying Referential Integrity Constraint	13
9.	Creating Sequence	14
10.	Creating Procedure	16
11.	After Inserting Values into the Tables	17
12.	Applying Trigger	19
13.	Query Outputs Based on Requirements	21
14.	Up to Which N.F	23

# 1. ABSTRACT

In this Courier Management System database .We begin our discussion of data base design starting with Requirement analysis, Conceptual data base design, Logical data base design and applying some operations on the database. The below lines describes the description of above mentioned topics clearly as per our knowledge. The below pages contains the information about the E.R diagrams, procedure and triggers and some operations according to the customer requirements.

## 2. Requirements of Customer

- ❖ According to the end user requirement the respective application has to calculate the cost of the courier with respect to the weight of the courier . The given cost of the courier having weight 1kg with courier type :
  - Express – 100 Rs
  - Ordinary – 80 Rs
- ❖ The maximum weight of the courier defined by the end user is 100 kg so, the end user asked that they should be alerted when they take more than 100kg weighted courier.
- ❖ The end user addressed a question that he want to know the total cost of the courier sent as per the day he required .
- ❖ The end user addressed a question that he want to know the total amount spending on employee salary .
- ❖ The end user addressed a question that he want to know that number of couriers send to particular area on particular day.

### 3 .Entities of the database

An **Entity** is an object in the real world that is distinguishable from other objects. An Entity is described using set of attributes. It is often useful to identify a collection of similar entities. such collection is called an **entity set**. All entities in a given entity set have the same attributes.

● The Application Courier Management system consists of entities :

1. Office
2. Employee
3. Courier
4. Customer
5. Transportation

➤ The relation **Office** has attributes :

1. office\_id - PRIMARY KEY
2. office\_name
3. office\_tel\_no
4. office\_desc

➤ The relation **Employee** has attributes :

1. emp\_id - PRIMARY KEY
2. emp\_name
3. emp\_type
4. emp\_ssn
5. emp\_tel\_no
6. emp\_sal

➤ The relation **Courier** has attributes :

1. c\_id - PRIMARY KEY
2. c\_desc
3. c\_pieces
4. c\_type
5. c\_weight
6. c\_cost
7. c\_time
8. c\_date
9. c\_from
10. c\_to

➤ The relation **Customer** has attributes :

1. Cust\_ssn - PRIMARY KEY
2. Cust\_name
3. Cust\_phno
4. Cust\_addr

➤ The relation **Transportation** has attributes :

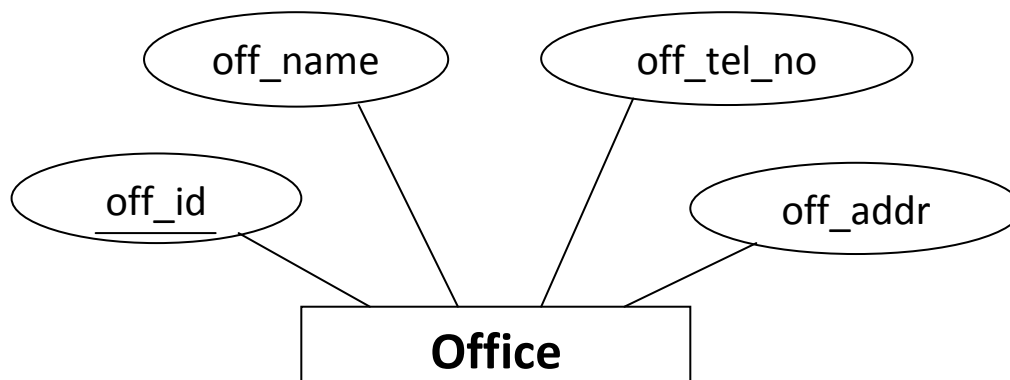
1. Driver\_id - PRIMARY KEY
2. Vehicle\_no
3. Dep\_time
4. Reached\_time
5. Status

## 4 .E-R Diagrams of the Database

**Entity Relationship Diagrams:** The *entity –relationship data model* allows us to describe the data involved in a real world enterprise in terms of objects and their relationships and is widely used to develop an initial database design. It provides useful concepts that allow us to from an informal description of what users want from their database to a more detailed, precise description that can be implemented in a DBMS.

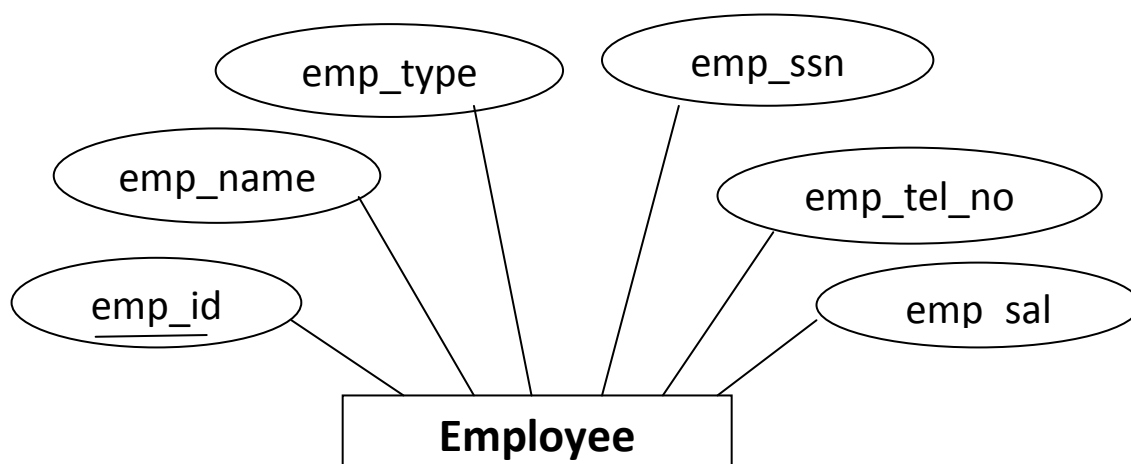
The relevant information about the **Courier Management System** application Is given below:

The E-R Diagram of the entity **office** is :



**fig : 4.1**

The E-R Diagram of the entity **Employee** is :



**fig : 4.2**

The E-R Diagram of the entity **Courier** is :

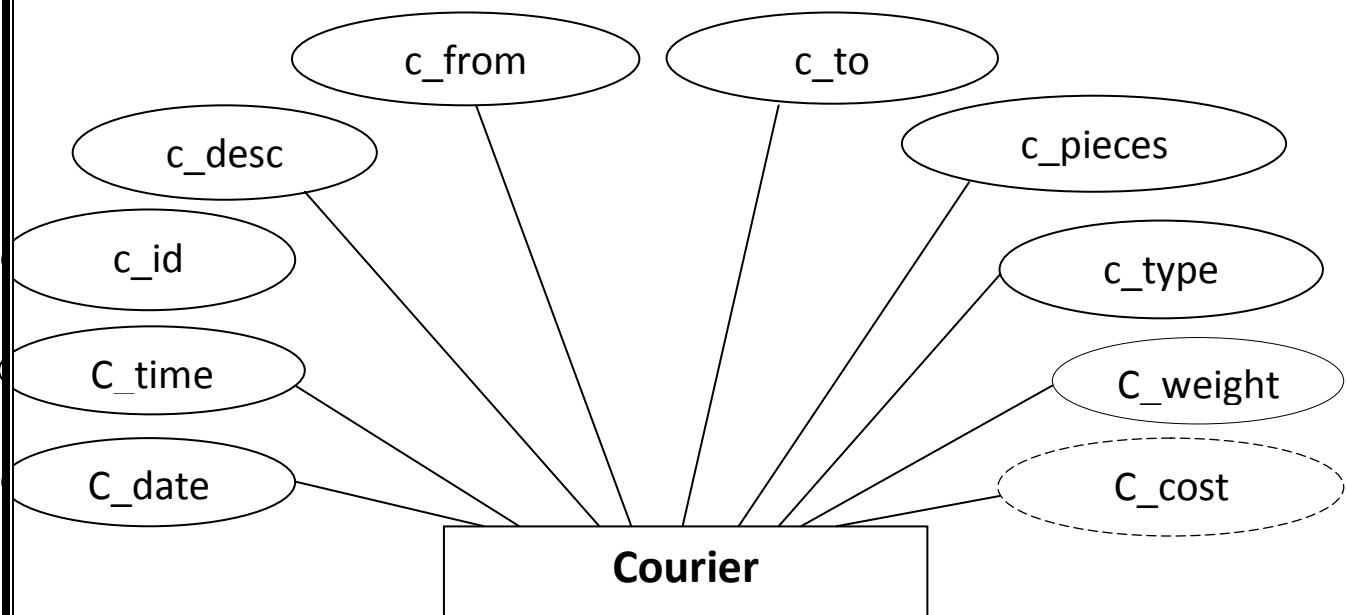


fig : 4.3

The E-R Diagram of the entity **Customer** is :

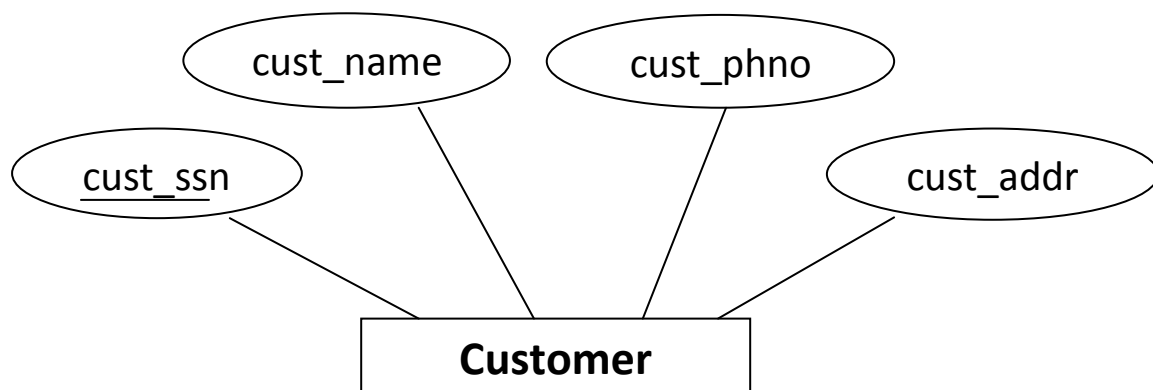
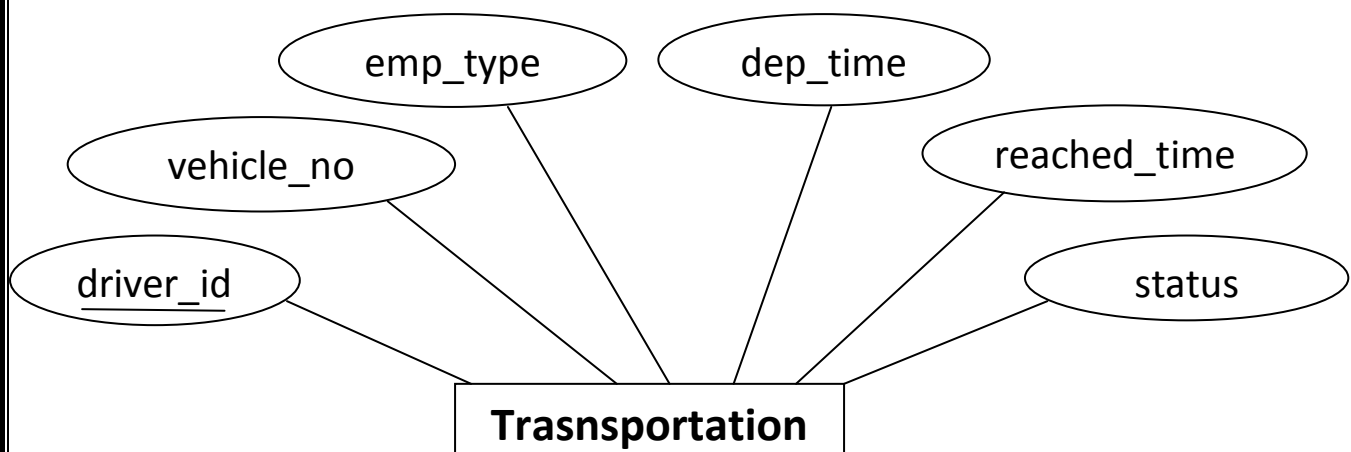


fig: 4.4

The E-R Diagram of the entity **Transportation** is :





## 5 .Total E-R diagram of Courier management system database

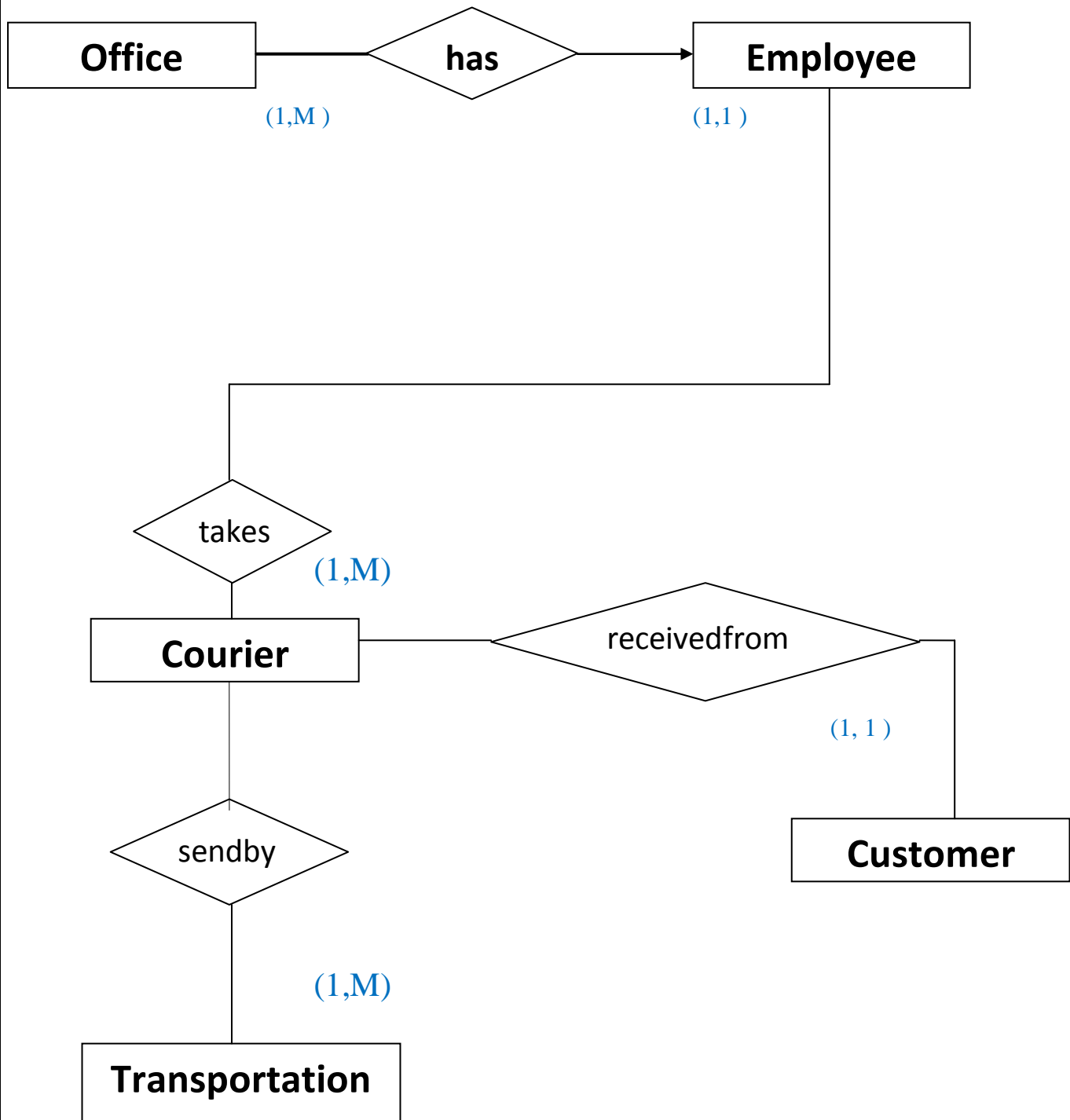


fig : 5.1

## 6. Schema of the database

- **Office** (off\_id: varchar,off\_name:varchar,off\_tel\_no:number, off\_desc :varchar)
- **Employee** (emp\_id:varchar,emp\_name:varchar,emp\_type :varchar,emp\_ssn:number,emp\_tel\_no:number,emp\_sal :number)
- **Courier**(c\_id:varchar,c\_desc:varchar,c\_from:varchar,c\_to:varchar, c\_pieces:number,c\_type:varchar,c\_weight:number,c\_cost:real, c\_time:varchar,c\_date:date)
- **Customer** (cust\_ssn:varchar,cust\_name:varchar,cust\_phno:number, cust\_addr:varchar)
- **Trnsportation**(driver\_id:vachar,vehicle\_no:number,dep\_time:varchar, reached\_time:varchar,status:varchar)

## 7. Creating Tables

*Syntax:*

```
CREATE TABLE tablename (  
    column1 data_type [constraint] [,  
    column2 data_type [constraint] ] [,  
    PRIMARY KEY (column1 [, column2]) ] [,  
    FOREIGN KEY (column1 [, column2])  
    REFERENCES tablename] [,CONSTRAINT  
    constraint]);
```

\*To create relation 'OFFICE' :

```
CREATE TABLE office(  
    off_id varchar(10) PRIMARY KEY,  
    off_name varchar(22) ,  
    off_tel_no number(10),  
    off_desc varchar(50) );
```

\*To create relation 'EMPLOYEE' :

```
CREATE TABLE employee(  
    emp_id varchar(10) PRIMARY KEY,  
    emp_name varchar(25) ,  
    emp_type varchar(12) NOT NULL,  
    emp_ssn number(10) ,  
    emp_tel_no number(10) ,  
    emp_sal number(10) );
```

\*To create relation 'COURIER' :

```
CREATE TABLE courier ( c_id    varchar (10) PRIMARY KEY,  
                        c_desc   varchar(501),  
                        c_from   varchar(70),  
                        c_to     varchar(70) NOT NULL,  
                        c_pieces number(5)  NOT NULL,  
                        c_type   varchar(15),  
                        c_weight number(10,2) NOT NULL,  
                        c_cost   number(8,2),  
                        c_time   varchar(12),  
                        c_date   date );
```

\*To create relation 'CUSTOMER' :

```
CREATE TABLE customer(  
    cust_ssn   varchar(10)  PRIMARY KEY ,  
    cust_name  varchar(25)  NOT NULL,  
    cust_phno  number(10),  
    cust_addr  varchar(50) );
```

\*To create relation 'TRANSPORTATION' :

```
CREATE TABLE transportation (  
    driver_id   varchar(10) PRIMARY KEY,  
    vehicle_no  number(8),  
    dep_time    varchar(8),  
    reached_time varchar(8),  
    status      varchar(50) );
```

## 8. Applying Referential Integrity Constraints

- ❖ The attributes emp\_offid, c\_custssn , c\_empid , c\_driverid are created using ALTER command. Then applying referential integrity constraints.

➤ ALTER TABLE employee

```
ADD CONSTRAINT emp_fk_offid FOREIGN KEY(emp_offid)
REFERENCES OFFICE ;
```

➤ ALTER TABLE courier

```
ADD CONSTRAINT c_fk_emp FOREIGN KEY(c_empid)
REFERENCES employee
```

```
ADD CONSTRAINT c_fk_custssn FOREIGN KEY(c_custssn)
REFERENCES customer
```

```
ADD CONSTRAINT c_fk_driverid FOREIGN KEY(c_driverid)
REFERENCES transportation;
```

## 9. Creating sequences

### SEQUENCES (AUTONUMBER):

A sequence is an object in Oracle that is used to generate a number sequence. This can be useful when you need to create a unique number to act as a primary key.

#### Syntax:

```
CREATE SEQUENCE sequence_name  
MINVALUE      value  
MAXVALUE      value  
START WITH POSITIVE_NUMBER  
INCREMENT BY POSITIVE_NUMBER  
[CACHE value |NOCACHE];
```

#### Operations on sequences:

**sequence\_name.NEXTVAL** – To retrieve next value from the sequence object.

**sequence\_name.CURRVAL** – To retrieve current value from the sequence object.

#### To Drop the sequence:

##### Syntax:

```
DROP SEQUENCE sequence_name;
```

- The user sequences in this database are :
  1. offid\_seq
  2. empid\_seq
  3. cid\_seq
  4. driverid\_seq

- To create sequence offid\_seq :

```
CREATE SEQUENCE offid_seq
```

```
start with 100
```

```
increment by 1 ;
```

- To create sequence empid\_seq :

```
CREATE SEQUENCE empid_seq
```

```
start with 1000
```

```
increment by 1 ;
```

- To create sequence cid\_seq :

```
CREATE SEQUENCE cid_seq
```

```
start with 500
```

```
increment by 1 ;
```

- ❖ To create sequence driverid\_seq :

```
CREATE SEQUENCE driverid_seq
```

```
start with 1200
```

```
increment by 1 ;
```

## 10. procedure to find the cost of the courier

A stored procedure is a named collection of procedural and SQL statements.

As per the end-user first requirement we written the procedure as follows :

**Syntax:**

```
CREATE OR REPLACE PROCEDURE procedure_name
[(argument [IN/OUT] DATA-TYPE . . .)]
[IS/AS] [variable_name datatype[:=initial_value]]
BEGIN
    PL/SAL or SQL statements;
    . . . . .
    . . . . .
END;
```

**Creating a Procedure: P\_COST**

```
create or replace procedure proc_cost
is
begin
update courier set c_cost=c_weight*100
where c_type='express';
update courier set c_cost=c_weight*50
where c_type='ordinary';
end;
/
```

To execute procedure : **exec p\_cost**



# 11. After Inserting the values into the tables

The resulting tables are :

## ❖ OFFICE :

OFF_ID	OFF_NAME	OFF_TEL_NO	OFF_DESC
100	ramnagar	7955443318	6-5-211,main branch,anantapur
101	marutinagar	7955453318	6-2-291,main branch,hyderabad
102	gandhinagar	7955453348	6-9-294,sub branch,anantapur
103	vidyuthnagar	7955453343	6-9-29,sub branch,hyderabad
104	tilaknagar	7955453311	6-9-277,main branch,bangalore
105	himayathnagar	7955453391	6-9-299,sub branch,bangalore

fig : 11.1

## ❖ EMPLOYEE :

EMP_ID	EMP_NAME	EMP_TYPE	EMP_SSN	EMP_TEL_NO	EMP_SAL	EMP_OFFID
1002	raj	departure staff	88883332222	9955443311	7000	100
1003	krishna	working staff	88883332223	9955443312	10000	101
1004	swathi	working staff	88883332225	9955443313	10000	102
1005	suda	departure staff	88883332227	9955443314	7000	103
1006	kamala	working staff	88883332228	9955443318	10000	104
1007	barathi	departure staff	888833352228	8955443318	7000	105

fig:11.2

## ❖ COURIER :

C_ID	C_DESC	C_FROM	C_TO	C_PIECES	C_TYPE	C_WEIGHT	C_COST	C_TIME	C_DATE	C_CUSTSSN	C_DRIVERID	C_EMPID
500	cpu	atp	bangalore	2	express	20	2000	9:50 am	29-JUN-17	68645533392	1200	1002
501	tv	atp	hyderabad	4	ordinary	40	3200	9:30 am	02-JUL-17	686455333916	1201	1003
502	refrigerator	bangalore	atp	3	express	90	9000	12:00 am	10-JUL-17	686455333915	1202	1004
503	novels	hyderabad	bangalore	100	express	50	5000	12:30 am	18-AUG-17	686455333917	1203	1005
504	dressess	atp	bangalore	10	ordinary	5	400	11:40 am	21-AUG-17	686455333918	1204	1006
505	jewellery	bangalore	hyderabad	10	express	1	100	11:40 am	21-AUG-17	686455333919	1205	1007
560	charger	kurnool	atp	1	express	11	1100	11:00 am	26-OCT-17	-	-	-
561	laptop	bangalore	atp	1	express	2	200	11:00 am	26-OCT-17	-	-	-
580	bicycle	hyderabad	atp	3	express	75	7500	11:00 am	26-OCT-17	-	-	-
581	bicycle	hyderabad	atp	3	express	75	7500	11:00 am	26-OCT-17	-	-	-

fig : 11.3

## ❖ CUSTOMER :

CUST_SSN	CUST_NAME	CUST_PHNO	CUST_ADDR
68645533392	thayyaba	9182029088	1-208,2nd road,anantapur,515001
686455333916	uma	8639881052	12-4-352,obuldevnagar,anantapur,515001
686455333915	tejashwini	9966087647	18-7-95,tilaknagar,guntakal,515801
686455333917	syam.kakarla	9603656549	6-5-349,maruthinagar,anantapur,515001
686455333918	umesh	9603656540	6-5-340,maruthinagar,anantapur,515001
686455333919	tarak	9603656541	6-5-34,maruthinagar,anantapur,515001
row(s) 1 - 6 of 6			

fig :11.4

## ❖ TRANSPORTATION :

DRIVER_ID	VEHICLE_NO	DEP_TIME	REACHED_TIME	STATUS
1200	ap02-512	9:40 am	10:40 am	delivered
1201	ap02-516	12:45 pm	10:40 am	hindupur
1202	ap02-312	1:00 pm	2:00 pm	delivered
1203	ap02-231	12:00 pm	1:00 pm	kurnool
1204	ap02-321	1:30 pm	2:30 pm	guntakal
1205	ap02-875	10:30 am	11:30 am	delivered
row(s) 1 - 6 of 6				

fig : 11.5

## 12. Applying Trigger

As per the end-user second requirement we written a trigger as follows:

**Trigger:** A trigger is procedural SQL code that is *automatically* invoked by the RDBMS upon the occurrence of a given data manipulation event.

There are two types of triggers:

- i. Statement-level trigger
  - ii. Row-level trigger
- i. A statement-level trigger is assumed if you omit the FOR EACH ROW keywords. This type of trigger is executed once, before or after the triggering statement is completed. This is the default case.
  - ii. A row-level trigger requires use of the FOR EACH ROW keywords. This type of trigger is executed once for each row affected by the triggering statement.

**Syntax :**

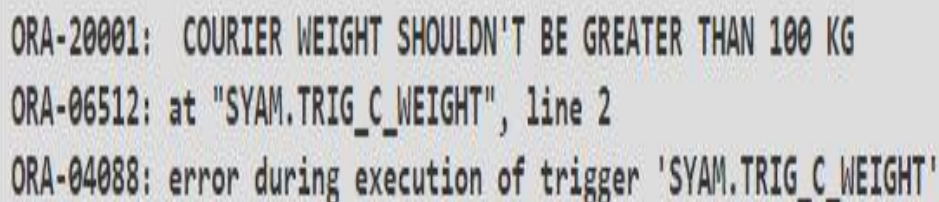
```
CREATE OR REPLACE TRIGGER trigger_name
[BEFORE/AFTER] [DELETE/INSERT/UPDATE OF
column_name] ON table_name
[FOR EACH ROW]
[DECLARE]
    [variable_named date type[:=initial_value]]
BEGIN
    PL/SQL insrtuitions;
    .....
END;
```

- To place a trigger when courier weight is greater than 100 kg.

```
CREATE OR REPLACE TRIGGER trig_c_weightT
BEFORE INSERT OR UPDATE OF c_weight ON courier
FOR EACH ROW
WHEN (NEW.c_weight <= 100)
BEGIN
    RAISE_APPLICATION_ERROR (-20001, ' COURIER WEIGHT
    SHOULDN"T BE GREATER THAN 100 KG');
END;
/
```

### Output :

❖ The error raised by the trigger :

A screenshot of an Oracle database error log. It shows three lines of text: 'ORA-20001: COURIER WEIGHT SHOULDN'T BE GREATER THAN 100 KG', 'ORA-06512: at "SYAM.TRIG\_C\_WEIGHT", line 2', and 'ORA-04088: error during execution of trigger 'SYAM.TRIG\_C\_WEIGHT''. The text is in a monospaced font and is displayed on a light gray background.

```
ORA-20001: COURIER WEIGHT SHOULDN'T BE GREATER THAN 100 KG
ORA-06512: at "SYAM.TRIG_C_WEIGHT", line 2
ORA-04088: error during execution of trigger 'SYAM.TRIG_C_WEIGHT'
```

**fig : 12.1**

## 13. Query outputs based on requirements

Query to know the total cost of the courier as per the day :

\*As per the end-user third requirement we written a query as follows:

```
SELECT SUM(c_cost) AS total_cost
FROM courier
WHERE c_date = '26-OCT-17' );
```

**OUTPUT:**

TOTAL_COST
16300

1 rows returned in 0.01 seconds

**fig : 13.1**

\*The total amount spending on employee salary :

```
SELECT SUM(emp_sal) AS
TOTAL_SALARY_SPENDING_EMP
FROM employee ;
```

**OUTPUT:**

TOTAL_SALARY_SPENDING_EMP
51000

**fig : 13.2**

\*To know that number of couriers send to particular area on particular day:

```
SELECT COUNT( c_id ) AS TOTAL_SENT  
FROM courier  
WHERE c_to = 'atp' and c_date = '26-oct- 17' ;
```

OUTPUT:

TOTAL_SENT
4

1 rows returned in 0.02 seconds

**fig : 13.3**

## 14. Up to which NF

- ❖ The present database is Courier Management System , Here we are to state the normal form of the database .
  - Our database satisfies the **1NF** , The first normal states that every relation has only single valued attributes ie., atomic variables . since we have taken the address as atomic variable.
  - Our database satisfies the **2NF**, The second normal form states that every relation has to satisfy 1NF and every non key columns depend on a key but not the subset of the key. All the non-prime attributes are functionally depend to keys of each relation.
  - Our database satisfies the **3NF**, The third normal form states that every relation has to satisfy 2NF and every non-key doesn't depend on non-key column.
- ◆ Finally, our Courier Management System database is up to **3NF**

----- END -----