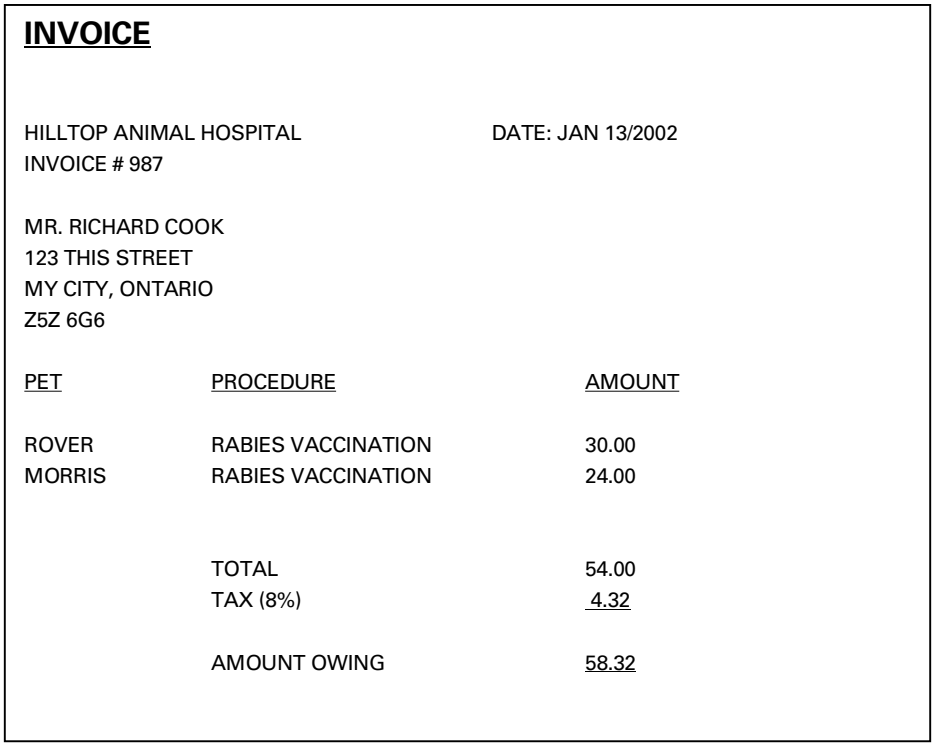
Data Normalization Example

The following image is an invoice of a pet hospital. This examples shows all the steps to normalize the data.



INVOICE

HILLTOP ANIMAL HOSPITAL — DATE: JAN 13/2002
INVOICE # 987

MR. RICHARD COOK
123 THIS STREET
MY CITY, ONTARIO
Z5Z 6G6

PET	PROCEDURE	AMOUNT
ROVER	RABIES VACCINATION	30.00
MORRIS	RABIES VACCINATION	24.00
	TOTAL	54.00
	TAX (8%)	4.32
	AMOUNT OWING	58.32

UNF
(Unnormalized Form):

- Identify all attributes presented in user view
- Choose a primary key (made up of 1 or more attributes) that best represents what user view describes
- Name the relation and list all attributes for the relation
- Indicate primary key by underlining 1 or more attributes
- Indicate if an attribute or group of related attributes can have more than one value for a given value of the primary key by enclosing within brace brackets { } – this is referred to as a 'repeating group'

invoice [invoice_no, invoice_date, cust_name, cust_addr, {pet_name, procedure, amount}]

1NF:

- A relation is in 1st normal form when the primary key determines a single value of each attribute for all attributes in the relation (i.e. the relation contains no repeating groups)
- Restate original unnormalized relation without repeating group:
  invoice [<u>invoice_no</u>, invoice_date, cust_name, cust_addr]

- Create new relation consisting of key of original relation and attributes within repeating group and add to key to ensure uniqueness:
  invoice_pet [<u>invoice_no</u>, <u>pet_id</u>, pet_name, procedure, amount]
  note: pet_id was chose as a key because pet_name is a character string and not a good key candidate.

- Restate the relation(s)

invoice [invoice_no, invoice_date, cust_name, cust_addr]

invoice_pet [invoice_no, pet_id, pet_name, procedure, amount]

2NF:

- A 1NF relation is in 2NF when the entire primary key is needed to determine the value of each non-key attribute (i.e. relation has no partial dependencies – attributes whose values can be determined by knowing only part of the key)

- 1NF Relations: **invoice_pet [<u>invoice_no</u>, <u>pet_id</u>, pet_name, procedure, amount]** contains the partial dependency **<u>pet_id</u>** -> **pet_name**

- Create new relation(s) consisting of part of the primary key and all attributes whose values are determined by this part of the primary key: **pet [<u>pet_id</u>, pet_name]**

- Restate original relation(s) without partially dependent attributes:

invoice [invoice_no, invoice_date, cust_name, cust_addr]

invoice_pet [invoice_no, pet_id, procedure, amount]

pet [pet_id, pet_name]

3NF:

- A 2NF relation is in 3NF when the primary key and nothing but the primary key can be used to determine the value of each non-key attribute (i.e. relation has no transitive dependencies – attributes whose values can be determined by knowing something other than the key)
  2NF Relations: **invoice [<u>invoice_no</u>, invoice_date, cust_name, cust_addr]**
  invoice_pet [<u>invoice_no</u>, <u>pet_id</u>, procedure, amount]
  pet [<u>pet_id</u>, pet_name]

- Create new relation(s) consisting of the attribute(s) which are determined by something other than the primary key (transitive dependencies) and make the primary key of these new relation(s) the attribute that actually determines the value of these attributes. In **invoice** the **cust_ addr** is determined by **cust_name** so create the new relation: **customer [<u>cust_no</u>, cust_name, cust_street, cust_city, cust_pstlcd]**
  note: cust_no was chose as a key because cust_name is a character string and not a good key candidate. The customer address was broken apart in 3NF. All foreign keys are identified.

- Restate original relation(s) without transitively dependent attributes (Original relation will now contain a foreign key – a non-key attribute that relates to the primary key of the new relation) :

invoice [invoice_no, invoice_date, cust_no (FK)]

invoice_pet [invoice_no (FK), pet_id (FK), procedure, amount]

pet [pet_id, pet_name]

customer [cust_no, cust_name, cust_street, cust_city, cust_pstlcd]