DBMS Project

Electricity Billing Database Management System

Pratik Patil - 202118023
Bhavya Chandrasala - 202118028
Jeffrey James - 202118031
Tanya Jagyasi - 202118039

M.Sc. Data Science
Dhirubhai Ambani Institute of Information and Communication Technology - Gandhinagar

Index

➢ Scope of Project:

- Stores and displays the records of each and every customer, connected to the electric grid, from their respective account.
- It tracks all the information related to Electricity Board and different areas coming under it.
- Helps manage the details of electricity bills in an area, supplier and their customer details, along with customer feedback.
- Easily accessible database connected to PostgreSQL for efficiently returning output of queries.

➢ Description:

The National Grid is the high-voltage electricity transmission network in India, connecting power stations and major substations, ensuring that electricity generated anywhere in India can be used to satisfy demand elsewhere.
Electric supply is done all over the country as electricity is a basic need for today's world. As a result, even in cities, a huge amount of data is generated by households that consume electricity for their daily needs.

Problem Statement:

Now a days people find it very difficult to keep the records of their electric bill payments. This system will help them to keep track of their payments. Problems like maintenance of Meters will also be considered which are neglected by most people.

In this project, we will be using database of a city to access information of the electricity usage in different areas and even particular households.
Every customer connected to the electricity board will have their accounts, which after signing in would provide details of their unique invoices and other individual information.

There are different methods of transactions in this modern life scenario which will be monitored in the billing entry, which will also include details of date and month of transaction.

Customer details such as:

- Customer ID
- Customer Name
- Address

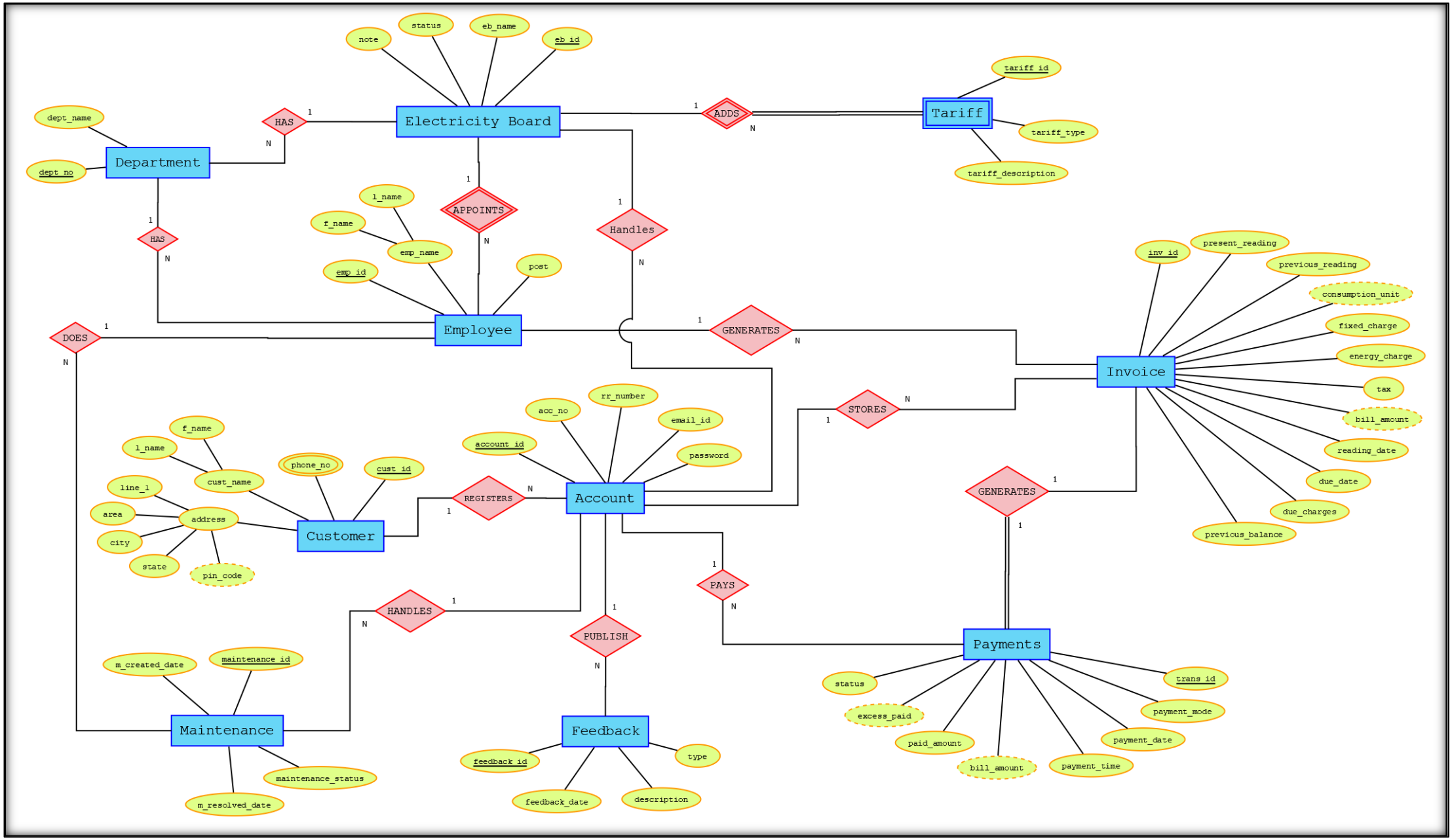
  Etc.

Along with other relevant information would be stored in different relations connected to each other.

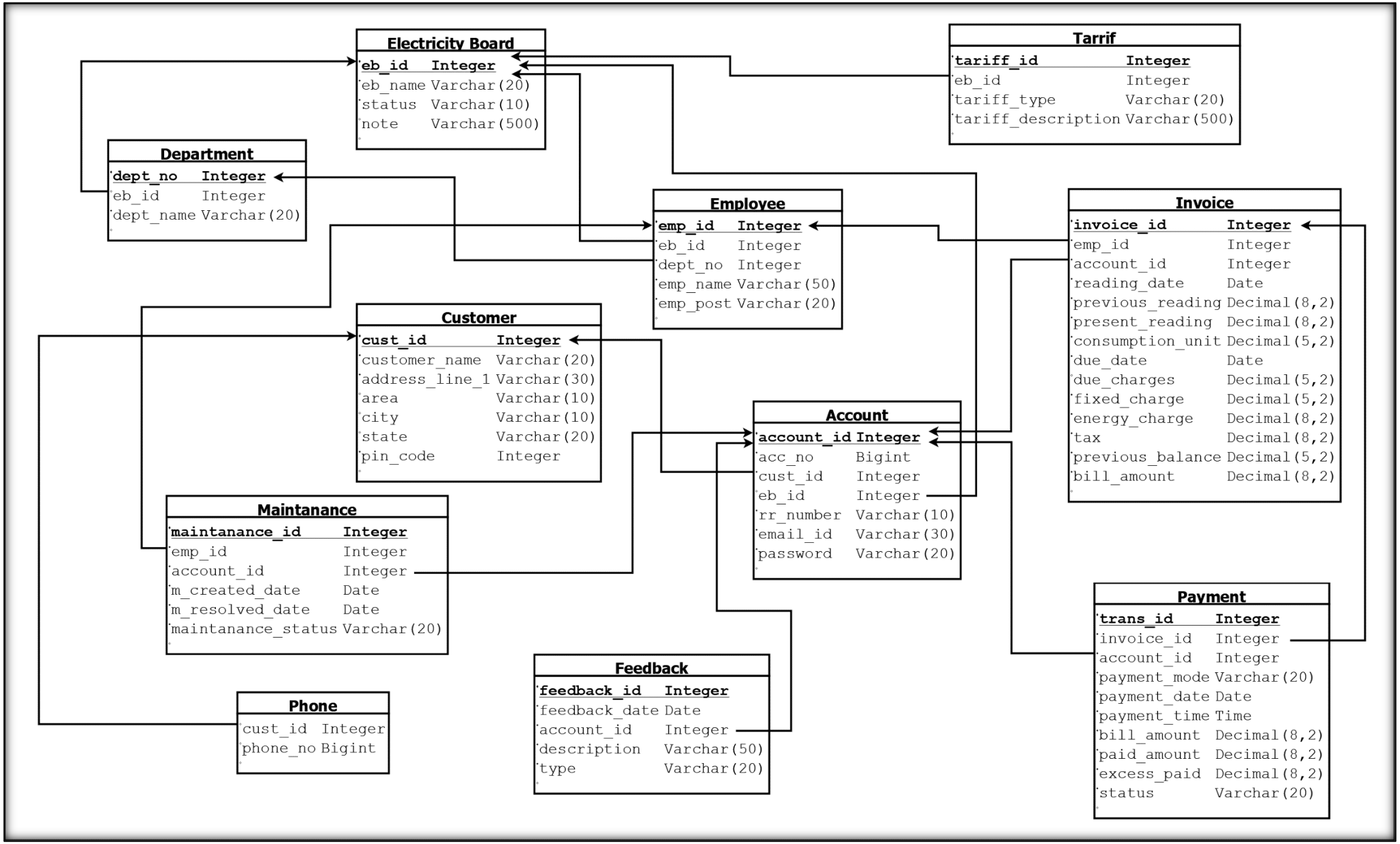
In this database, a given customer/household will have the customer details, their power consumption in units, electric bill. This primary entity which will be linked with other entities like Account, Billing, Feedback of those individual, which further relate with Invoices, Tariff and Electricity board and companies giving the power supply. The system can receive the customer’s monthly power consumption information which can be used to calculate the average power consumption of a household or even for a selected area.

This system will make this complex database easy to access from anywhere.

➢ Entity Relationship (ER) Diagram:



➢ Relational Schema Diagram:



➢ Functional Dependencies and Normalization:

1) Customer (cust_id, cust_name, address_line_1, area, city, state, pin_code)

This table is in 2NF.

{cust_id} -> cust_name

{cust_id} -> address_line_1

{cust_id} -> area

{cust_id} -> city

{cust_id} -> state

{cust_id} -> pin_code

{pin_code} -> area

{pin_code} -> city

{pin_code} -> state

Candidate Key: {cust_id}

Prime Attribute: cust_id, pin_code

Non-Prime Attribute: cust_name, address_line_1, area, city, state

Normalization to 3NF and BCNF:

pincode is not unique, thus it is in 2NF form. So, to convert it to BCNF cust_id and pincode will be together declared as a super key which will uniquely identify user city and user state.

{cust_id} -> cust_name

{cust_id} -> address_line_1

{cust_id, pin_code} -> area

{cust_id, pin_code} -> city

{cust_id, pin_code} -> state

2) Phone_no (cust_id, phone_no)

This table is in 3NF and BCNF.

{cust_id} -> phone_no

Candidate Key: {cust_id}

Prime Attribute: cust_id

Non-Prime Attribute: phone_no

Explanation:

A relation is in third normal form and BCNF, as there is no transitive dependency for non-prime attributes as well as it is in second normal form.

3) Electricity_board (eb_id, eb_name, status, note)

This table is in 3NF and BCNF.

{eb_id} -> eb_name

{eb_id} -> status

{eb_id} -> note

Candidate Key: {eb_id}

Prime Attribute: eb_id

Non-Prime Attribute: eb_name, status, note

Explanation:

A relation is in third normal form and BCNF, as there is no transitive dependency for non-prime attributes as well as it is in second normal form.

4) Tarrif (tarif_id, eb_id, tarrif_type, tarrif_description, tarrif_rate)

This table is in 3NF and BCNF.

{tarrif_id} -> tarrif_type

{tarrif_id} -> eb_id

{tarrif_id} -> tarrif_description

{tarrif_id} -> tarrif_rate

Candidate Key: {tarrif_id}

Prime Attribute: tarrif_id

Non-Prime Attribute: tarif_type, eb_id, tarrif_description, tarrif_rate

Explanation:

A relation is in third normal form and BCNF, as there is no transitive dependency for non-prime attributes as well as it is in second normal form.

5) Account (account_id, account_no, cust_id, eb_id, rr_no, email_id, password)

This table is in 3NF and BCNF.

{account_id} -> account_no

{account_id} -> cust_id

{account_id} -> eb_id

{account_id} -> rr_no

{account_id} -> email_id

{account_id} -> password

Candidate Key: {account_id}

Prime Attribute: account_id

Non-Prime Attribute: account_no, cust_id, rr_no, email_id, password

Explanation:

A relation is in third normal form and BCNF, as there is no transitive dependency for non-prime attributes as well as it is in second normal form.

6) Department (dept_no,eb_id, dept_name)

This table is in 3NF and BCNF.

{dept_no} -> eb_id

{dept_no} -> dept_name

Candidate Key: {dept_no}

Prime Attribute: dept_no

Non-Prime Attribute: eb_id**,** dept_name

Explanation:

A relation is in third normal form and BCNF, as there is no transitive dependency for non-prime attributes as well as it is in second normal form.

7) Employee (emp_id, eb_id, dept_no, emp_name, emp_post)

This table is in 3NF and BCNF.

{emp_id} -> eb_id

{emp_id} -> dept_no

{emp_id} -> emp_name

{emp_id} -> emp_post

Candidate Key: {emp_id}

Prime Attribute: emp_id

Non-Prime Attribute: eb_id, dept_no, emp_name, emp_post

Explanation:

A relation is in third normal form and BCNF, as there is no transitive dependency for non-prime attributes as well as it is in second normal form.

8) Invoice (invoice_id, emp_id, account_id, present_reading, reading_date, consumption_unit, previous_reading, previous_balance, fixed_charge, energy_charge, tax, bill_amount, due_date)

This table is in 3NF and BCNF.

{invoice_id} -> emp_id

{invoice_id} -> account_id

{invoice_id} -> present_reading

{invoice_id} -> reading_date

{invoice_id} -> consumption_unit

{invoice_id} -> previous_reading

{invoice_id} -> previous_balance

{invoice_id} -> fixed_charge

{invoice_id} -> energy_charge

{invoice_id} -> tax

{invoice_id} -> bill_amount

{invoice_id} -> due_date

Candidate Key: {invoice_id}

Prime Attribute: invoice_id

Non-Prime Attribute: emp_id, account_id, present_reading, reading_date, consumption_unit, previous_reading, previous_balance, fixed_charge, energy_charge, tax, bill_amount, due_date

Explanation:

A relation is in third normal form and BCNF, as there is no transitive dependency for non-prime attributes as well as it is in second normal form.

9) Payment (trans_id, invoice_id, account_id, payment_mode, payment_date, payment_time, bill_amount, paid_amount, excess_paid, status)

This table is in 3NF and BCNF.

{trans_id} -> invoice_id

{trans_id} -> account_id

{trans_id} -> payment_mode

{trans_id} -> payment_date

{trans _id} -> payment_time

{trans _id} -> bill_amount

{trans _id} -> paid_amount

{trans _id} -> excess_paid

{trans _id} -> status

Candidate Key: {trans_id}

Prime Attribute: trans_id

Non-Prime Attribute: invoice_id, account_id, payment_mode, payment_date, payment_time, bill_amount, paid_amount, excess_paid, status

Explanation:

A relation is in third normal form and BCNF, as there is no transitive dependency for non-prime attributes as well as it is in second normal form.

10) Feedback (feedback_id, feedback_date, account_id, description, type)

This table is in 3NF and BCNF.

{feedback_id} -> feedback_date

{feedback_id} -> account_id

{feedback_id} -> description

{feedback_id} -> type

Candidate Key: {feedback_id}

Prime Attribute: feedback_id

Non-Prime Attribute: feedback_date, account_id, description, type

Explanation:

A relation is in third normal form and BCNF, as there is no transitive dependency for non-prime attributes as well as it is in second normal form.

11) Maintenance (maintenance_id, emp_id, account_id, m_created_date, m_resolved_date, maintenance_status)

This table is in 3NF and BCNF.

{maintenance_id} -> emp_id

{maintenance_id} -> account_id

{maintenance_id} -> m_created_date

{maintenance_id} -> m_resolved_date

{maintenance_id} -> maintenance_status

Candidate Key: {maintenance_id}

Prime Attribute: maintenance_id

Non-Prime Attribute: emp_id, account_id, m_created_date, m_resolved_date, maintenance_status

Explanation:

A relation is in third normal form and BCNF, as there is no transitive dependency for non-prime attributes as well as it is in second normal form.

➢ DDL Script:

1. Creating Schema: Bill_Management

```
create schema Bill_Management;
set search_path to Bill_Management;
```

2. Creating Table: customer

```
create table customer(cust_id int PRIMARY KEY,
cust_name varchar(20),
address_line_1 varchar(30),
area varchar(10),
city varchar(10),
state varchar(20),
pin_code int);
```

3. Creating Table: phone_no

```
create table phone_no(cust_id int,
phone_no bigint,
FOREIGN KEY (cust_id) REFERENCES customer(cust_id) ON DELETE CASCADE ON UPDATE CASCADE);
```

4. Creating Table: electricity_board

```
create table electricity_board(eb_id int PRIMARY KEY,
eb_name varchar(20),
status varchar(10),
note varchar(500));
```

5. Creating Table: tariff

```
create table tariff(tarrif_id int PRIMARY KEY,
eb_id int,
tariff_type varchar(20),
tariff_description varchar(500),
FOREIGN KEY (eb_id) REFERENCES electricity_board(eb_id) ON DELETE CASCADE ON UPDATE CASCADE);
```

6. Creating Table: account

```
create table account(account_id int PRIMARY KEY,
acc_no bigint Unique,
cust_id int,
eb_id int,
rr_number varchar(10),
email_id varchar(30),
password varchar(20),
FOREIGN KEY (cust_id) REFERENCES customer(cust_id) ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (eb_id) REFERENCES electricity_board(eb_id) ON DELETE CASCADE ON UPDATE
CASCADE);
```

7. Creating Table: department

```
create table department(dept_no int PRIMARY KEY,
eb_id int,
dept_name varchar(20));
```

8. Creating Table: employee

```
create table employee(emp_id Int PRIMARY KEY,
eb_id int,
dept_no int,
emp_name varchar(50),
emp_post varchar(20),
FOREIGN KEY (eb_id) REFERENCES electricity_board(eb_id) ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (dept_no) REFERENCES department(dept_no) ON DELETE CASCADE ON UPDATE
CASCADE);
```

9. Creating Table: invoice

```
create table invoice(invoice_id int PRIMARY KEY,
emp_id int,
account_id int,
reading_date date,
previous_reading decimal(8,2,
present_reading decimal(8,2),
consumption_unit decimal(5,2),
due_date date,
due_charges decimal(5,2),
fixed_charge decimal(5,2),
energy_charge decimal(8,2),
tax decimal(8,2),
previous_balance decimal(5,2),
bill_amount decimal(8,2),
FOREIGN KEY (emp_id) REFERENCES employee(emp_id) ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (account_id) REFERENCES account(account_id) ON DELETE CASCADE ON UPDATE
CASCADE);
```

10. Creating Table: payment

```
create table payment(trans_id int PRIMARY KEY,
invoice_id int,
account_id int,
payment_mode varchar(20),
payment_date date,
payment_time time,
bill_amount decimal(8,2),
paid_amount decimal(8,2),
excess_paid decimal(8,2),
status varchar(20),
FOREIGN KEY (invoice_id) REFERENCES invoice(invoice_id) ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (account_id) REFERENCES account(account_id) ON DELETE CASCADE ON UPDATE
CASCADE);
```

11. Creating Table: feedback

```
create table feedback(feedback_id int PRIMARY KEY,
feedback_date date,
account_id int,
description varchar(50),
type varchar(20),
FOREIGN KEY (account_id) REFERENCES account(account_id) ON DELETE CASCADE ON UPDATE CASCADE);
```

12. Creating Table: maintenance

```
create table maintenance(maintenance_id int PRIMARY KEY,
emp_id int,
account_id int,
m_created_date date,
m_resolved_date date,
maintenance_status varchar(20),
FOREIGN KEY (emp_id) REFERENCES employee(emp_id) ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (account_id) REFERENCES account(account_id) ON DELETE CASCADE ON UPDATE CASCADE);
```

➢ DML Script (Insert):

1. Inserting Values in 'customer' Table:

insert into customer values(101,'Abdul Parmar','782/2,Jaypopat Apartments','Andheri','Mumbai','Maharashtra',400047)

insert into customer values(102,'Shreyas Patel','21, Gangutai Apartments','Bandra','Mumbai','Maharashtra',400050)

insert into customer values(103,'Ashok Dinda','12, Suryabhai Bungalows','Borivali','Mumbai','Maharashtra',400091)

insert into customer values(104,'Justin Christian','20, christio flats','Dahisar','Mumbai','Maharashtra',400068)

insert into customer values(105,'Kailash Mehta','10, Suryabhai Bungalows','Borivali','Mumbai','Maharashtra',400091)

insert into customer values(106,'Jessica Jones','768,Jaypopat Apartments','Andheri','Mumbai','Maharashtra',400047)

insert into customer values(107,'Luke Cage','24, christio flats','Dahisar','Mumbai','Maharastra',400068)

insert into customer values(108,'Harisingh Mehta','34,Lalubhai Society','Andheri','Mumbai','Maharashtra',400047)

insert into customer values(109,'Paul Matthew','12, Gokuldam society','Dahisar','Mumbai','Maharashtra',400068)

insert into customer values(110,'Azzarhudin Gariwala','2, madhuram flats','Borivali','Mumbai','Maharashtra',400091)

insert into customer values(111,'Avneet Singh','12, Gangutai Apartments','Bandra','Mumbai','Maharashtra',400050)

insert into customer values(112,'Shivani Rathi','12, madhuram flats','Borivali','Mumbai','Maharashtra',400091)

insert into customer values(113,'Bhavya Chandrasala','9, Gokuldam society','Dahisar','Mumbai','Maharashtra',400068)

insert into customer values(114,'Pratik Patil','17, Lalubhai society','Andheri','Mumbai','Maharashtra',400048)

insert into customer values(115,'Tanya Jagyasi','9, Gangutai Apartment','Bandra','Mumbai','Maharashtra',400057)

2. Inserting Values in ‘phone_no’ Table:

insert into phone_no values(101,9584548115)

insert into phone_no values(106,8468514124)

insert into phone_no values(108,8353281323)

insert into phone_no values(101,9745864648)

insert into phone_no values(114,9432134521)

insert into phone_no values(107,9213542121)

insert into phone_no values(107,9998654656)

insert into phone_no values(120,9732135435)

insert into phone_no values(102,9845121542)

insert into phone_no values(111,9755823543)

insert into phone_no values(112,9321321254)

insert into phone_no values(119,9743132154)

insert into phone_no values(104,9154134212)

insert into phone_no values(112,8786546854)

insert into phone_no values(113,9731834515)

3. Inserting Values in ‘electricity_board’ Table:

insert into electricity_board values(101010,'MSEB','active','Maharashtra State Electricity Board. Headquarters: Mumbai, Maharashtra, India. Owner: Government of Maharashtra Organization. Type: State-owned enterprise Area. Purpose: Power distribution.')

4. Inserting Values in ‘tariff’ Table:

insert into tariff values(111111,101010,'Residential','Units(0-100): ₹3.30, Units(101-300): ₹7.30, Units(301-500): ₹9.90, Units(>500): ₹11.50')

5. Inserting Values in ‘account’ Table:

insert into account
values(200001,231485762,101,101010,30000782,'abdulparmar@gmail.com','Lala@4527')

insert into account
values(200002,231485763,102,101010,30000784,'shreyaspatel@gmail.com','janeman$127')

insert into account
values(200003,231485764,103,101010,30000786,'ashokdinda@gmail.com','Mahism*739')

insert into account
values(200004,231485765,104,101010,30000788,'justinchrist@yahoo.com','Ranu@dola12')

insert into account
values(200005,231485766,105,101010,30000790,'kailashmehta@gmail.com','MrMehta*49')

insert into account
values(200006,231485767,106,101010,30000792,'jessjones@gmail.com','Jessica@137')

insert into account
values(200007,231485768,107,101010,30000794,'lukibaba@gmail.com','BabaLuke@21')

insert into account
values(200008,231485769,108,101010,30000796,'harisingh.mehta@yahoo.com','Gujujay*134')

insert into account
values(200009,231485770,109,101010,30000798,'paulmatthews@yahoo.com','Paul3719')

insert into account
values(200010,231485771,110,101010,30000800,'azzarbhai@gmail.com','Cricket1392')

insert into account
values(200011,231485772,111,101010,30000802,'avneetsingh@yahoo.com','lassi193')

insert into account
values(200012,231485773,112,101010,30000804,'rathishivani@gmail.com','rathi1421')

insert into account
values(200013,231485774,113,101010,30000806,'bhavyachand@yahoo.com','Mumbaiaaya131')

insert into account
values(200014,231485775,114,101010,30000808,'pratikpatil@gmail.com','nashikjan7492')

insert into account
values(200015,231485776,115,101010,30000810,'tanyajagyasi@yahoo.com','Samosa#143')

6. Inserting Values in ‘department’ Table:

insert into department values(1,101010,'Administration')

insert into department values(2,101010,'Technical')

7. Inserting Values in ‘employee’ Table:

insert into employee values(390001,101010,1,'Tipendar Gada','Station Operator')

insert into employee values(390002,101010,2,'Sundar Lal','Engineer')

insert into employee values(390003,101010,2,'Jethalal Gada','Engineer')

insert into employee values(390004,101010,2,'Hansraj Baldevraj','Wireman')

insert into employee values(390005,101010,2,'Popatlal Bhatia','Wireman')

insert into employee values(390006,101010,2,'Ramubhai Patel','Wireman')

insert into employee values(390007,101010,2,'Champaklal Gada','Wireman')

insert into employee values(390008,101010,1,'Roshan Singh','Meter Reader')

insert into employee values(390009,101010,1,'Bagheshwar Undhaiwala','Meter Reader')

insert into employee values(390010,101010,1,'Tarak Mehta','Meter Reader')

insert into employee values(390011,101010,1,'Aatmaram Bhide','Meter Reader')

8. Inserting Values in ‘invoice’ Table:

insert into invoice values(6001,390008,200001,'01-07-2021',13254,13594,340.00,'15-07-2021',0,150,2465,119,0,2734)

insert into invoice values(7001,390008,200001,'01-09-2021',13594,13894,300.00,'15-09-2021',0,150,2175,105,0,2430)

insert into invoice values(8001,390008,200001,'01-11-2021',13894,14010,116,'15-11-2021',0,150,841,40.6,0,1031.6)

insert into invoice values(6002,390009,200002,'01-07-2021',12514,12754,240.00,'15-07-2021',17.4,150,1740,84,0,1991.4)

insert into invoice values(7002,390009,200002,'01-09-2021',12754,12980,226.00,'15-09-2021',0,150,1638.5,79.1,0,1867.6)

insert into invoice values(8002,390009,200002,'01-11-2021',12980,13240,260,'15-11-2021',0,150,1885,91,0,2126)

insert into invoice values(6003,390010,200003,'01-07-2021',13004,13248,244.00,'15-07-2021',0,150,1769,97.6,0,2016.6)

insert into invoice values(7003,390010,200003,'01-09-2021',13248,13584,336,'15-09-2021',0,150,2436,134.4,0,2720.4)

insert into invoice values(8003,390010,200003,'01-11-2021',13584,13798,214,'15-11-2021',0,150,1551.5,85.6,0,1787.1)

insert into invoice values(6004,390011,200004,'05-07-2021',17815,18009,194,'20-07-2021',0,150,1406.5,77.6,0,1634.1)

insert into invoice values(7004,390011,200004,'05-09-2021',18009,18354,345,'20-09-2021',0,150,2501.25,138,0,2789.25)

insert into invoice values(8004,390011,200004,'05-11-2021',18354,18831,477,'20-11-2021',0,150,3458.25,190.8,0,3799.05)

insert into invoice values(6005,390010,200005,'03-07-2021',14945,15110,165,'18-07-2021',0,150,1196.25,66,0,1412.25)

insert into invoice values(7005,390010,200005,'03-09-2021',15110,15348,238,'18-09-2021',0,150,1725.5,95.2,0,1970.7)

insert into invoice values(8005,390010,200005,'03-11-2021',15348,15584,236,'18-11-2021',0,150,1711,94.4,0,1955.4)

9. Inserting Values in ‘payment’ Table:

Insert into payment values(6969121,6001,200001,'online','02-07-2021','11:45',2734,2734,0,'paid');

Insert into payment values(6969122,7001,200001,'','',,2430,0,'unpaid');

Insert into payment values(6969101,8001,200001,'','',,1032,0,'unpaid');

Insert into payment values(6969123,6002,200002,'online','18-07-2021','16:35',1991,1991,0,'paid');

Insert into payment values(6969124,7002,200002,'online','05-09-2021','08:05',1868,1868,0,'paid');

Insert into payment values(6969102,8002,200002,'','',,2126,0,'unpaid');

Insert into payment values(6969125,6003,200003,'online','12-07-2021','09:54',2017,2017,0,'paid');

Insert into payment values(6969126,7003,200003,'','',,2720,0,'unpaid');

Insert into payment values(6969103,8003,200003,'online','07-11-2021','15:57',1787,1800,13,'paid');

Insert into payment values(6969127,6004,200004,'online','09-07-2021','09:24',1634,1634,0,'paid');

Insert into payment values(6969128,7004,200004,'cash/cheque','07-09-2021','22:54',2789,2789,0,'paid');

Insert into payment values(6969104,8004,200004,'','',,3799,0,'unpaid');

Insert into payment values(6969129,6005,200005,'online','15-07-2021','20:54',1412,1412,0,'paid');

Insert into payment values(6969130,7005,200005,'cash/cheque','15-09-2021','18:24',1971,1971,0,'paid');

Insert into payment values(6969105,8005,200005,'','',,1955,0,'unpaid');

10. Inserting Values in ‘feedback’ Table:

```
insert into feedback values(2021101,'03-11-2021',200001,'Good service','Excellent')
insert into feedback values(2021102,'02-07-2021',200001,'Not satisfied','Poor')
insert into feedback values(2021103,'09-09-2021',200001,'Good service','Excellent')
insert into feedback values(2021104,'18-07-2021',200002,'','Good')
insert into feedback values(2021105,'05-09-2021',200002,'Good service','Excellent')
insert into feedback values(2021106,'04-11-2021',200002,'','Good')
insert into feedback values(2021107,'05-11-2021',200003,'','Good')
insert into feedback values(2021108,'09-07-2021',200004,'','Excellent')
insert into feedback values(2021109,'02-09-2021',200004,'','Excellent')
insert into feedback values(2021110,'15-07-2021',200005,'Not Satisfied','Poor')
insert into feedback values(2021111,'24-11-2021',200006,'','Good')
insert into feedback values(2021112,'05-09-2021',200007,'','Excellent')
insert into feedback values(2021113,'24-11-2021',200007,'Good service','Excellent')
insert into feedback values(2021114,'08-07-2021',200007,'','Excellent')
insert into feedback values(2021115,'05-07-2021',200008,'','Good')
```

11. Inserting Values in ‘maintenance’ Table:

```
insert into maintenance values(9601,390004,200001,'12-07-2021','20-07-2021','completed')
insert into maintenance values(9621,390004,200001,'12-10-2021','20-10-2021','completed')
insert into maintenance values(9602,390005,200002,'12-07-2021','20-07-2021','completed')
insert into maintenance values(9622,390005,200002,'12-10-2021','20-10-2021','completed')
insert into maintenance values(9603,390006,200003,'18-07-2021','30-07-2021','completed')
insert into maintenance values(9623,390006,200003,'18-10-2021','','pending');
insert into maintenance values(9604,390007,200004,'18-07-2021','30-07-2021','completed');
insert into maintenance values(9624,390007,200004,'18-10-2021','','pending');
insert into maintenance values(9605,390006,200005,'18-07-2021','30-07-2021','completed');
insert into maintenance values(9625,390006,200005,'18-10-2021','','pending');
insert into maintenance values(9606,390004,200006,'12-07-2021','20-07-2021','completed');
insert into maintenance values(9626,390004,200006,'12-10-2021','','pending');
insert into maintenance values(9607,390007,200007,'18-07-2021','02-08-2021','completed');
insert into maintenance values(9627,390007,200007,'18-10-2021','30-10-2021','completed');
insert into maintenance values(9608,390004,200008,'12-07-2021','22-07-2021','completed');
```

➢ Sample Queries and Results:

1. Retrieve all the customer whose Payment status is still pending.

Query:

select distinct customer.* from customer natural join account natural join payment where payment.status = 'unpaid' order by cust_id;

Output:

ebms/postgres@DBMS_Project

Query Editor | Query History | Scratch Pad

```
select distinct customer.* from customer natural join account natural join
payment where payment.status = 'unpaid' order by cust_id;
```

Data Output | Explain | Messages | Notifications

	cust_id [PK] integer	cust_name character varying (20)	address_line_1 character varying (30)	area character varying (10)	city character varying (10)	state character varying (10)	pin_code integer
1	101	Abdul Parmar	782/2,Jaypopat Apartments	Andheri	Mumbai	Maharastra	400047
2	102	Shreyas Patel	21, Gangutai Apartments	Bandra	Mumbai	Maharastra	400050
3	103	Ashok Dinda	12, Suryabhai Bungalows	Borivali	Mumbai	Maharastra	400091
4	104	Justin Christian	20, christio flats	Dahisar	Mumbai	Maharastra	400068
5	105	Kalish Mehta	10, Suryabhai Bungalows	Borivali	Mumbai	Maharastra	400091
6	106	Jessica Jones	768,Jaypopat Apartments	Andheri	Mumbai	Maharastra	400047
7	108	Harisingh Mehta	34, Lalubhai Society	Andheri	Mumbai	Maharastra	400047
8	109	Paul Matthew	12,Gokuldam society	Dahisar	Mumbai	Maharastra	400068
9	111	Avneet Singh	12,Gangutai Apartments	Bandra	Mumbai	Maharastra	400050
10	113	Bhavya Chandrasala	9, Gokuldam society	Dahisar	Mumbai	Maharastra	400068
11	114	Pratik Patil	17, Lalubhai Society	Andheri	Mumbai	Maharastra	400047
12	115	Tanya Jagyasi	9, Gangutai Apartments	Bandra	Mumbai	Maharastra	400050
13	116	Avi Kothari	24, madhuram Flats	Borivali	Mumbai	Maharastra	400091
14	118	Vatsal Desai	23, christio flats	Borivali	Mumbai	Maharastra	400091
15	119	Swati Singh	34, Mannat Bungalows	Bandra	Mumbai	Maharastra	400050
16	120	Yuvraj Singh	10,Lalubhai Society	Andheri	Mumbai	Maharastra	400047

2. Check the Payment status of customer Tanya Jagyasi.

Query:

select payment.payment_date,payment.status from payment natural join account natural join customer where cust_name = 'Tanya Jagyasi';

Output:

ebms/postgres@DBMS_Project

Query Editor | Query History

```
select payment.payment_date,payment.status from payment natural join
account natural join customer where cust_name = 'Tanya Jagyasi';
```

Data Output | Explain | Messages | Notifications

	payment_date date	status character varying (20)
1	2021-07-02	paid
2	2021-09-02	paid
3	[null]	unpaid

3. Retrieve all the customers (cust_id, cust_name) from Andheri having an electricity bill between 1500 to 2500 for September month.

Query:

select customer.cust_id,customer.cust_name from customer natural join account natural join invoice where area = 'Andheri' and bill_amount between 1500 and 2500 and due_date >= '01-09-2021' and due_date <= '30-09-2021';

Output:

ebms/postgres@DBMS_Project

Query Editor | Query History

```
select customer.cust_id,customer.cust_name from customer natural join account
natural join invoice where area = 'Andheri' and bill_amount
between 1500 and 2500 and due_date >= '01-09-2021' and due_date <= '30-09-2021';
```

Data Output | Explain | Messages | Notifications

	cust_id [PK] integer	cust_name character varying (20)
1	101	Abdul Parmar
2	106	Jessica Jones
3	108	Harisingh Mehta
4	114	Pratik Patil
5	120	Yuvraj Singh

4. **List top 10 customers (cust_id, cust_name, consumption_unit, billing_amount) having the highest billing amount.**

Query:

select customer.cust_id,customer.cust_name,invoice.consumption_unit,invoice.bill_amount from customer natural join account natural join invoice order by bill_amount desc limit 10;

Output:

ebms/postgres@DBMS_Project

Query Editor | Query History

```
select customer.cust_id,customer.cust_name,invoice.consumption_unit,invoice.bill_amount
from customer natural join account natural join invoice order by bill_amount desc limit 10;
```

Data Output | Explain | Messages | Notifications

	cust_id integer	cust_name character varying (20)	consumption_unit numeric (5,2)	bill_amount numeric (8,2)
1	104	Justin Christian	477.00	3799.05
2	113	Bhavya Chandrasala	409.00	3278.85
3	118	Vatsal Desai	360.00	2904.00
4	111	Avneet Singh	346.00	2804.69
5	104	Justin Christian	345.00	2789.25
6	115	Tanya Jagyasi	346.00	2779.60
7	101	Abdul Parmar	340.00	2734.00
8	103	Ashok Dinda	336.00	2720.40
9	120	Yuvraj Singh	326.00	2667.54
10	112	Shivani Rathi	326.00	2643.90

5. **List customers (cust_id, cust_name) with total bill Amount > 2000 and are still left to pay the bill.**

Query:

select customer.cust_id,customer.cust_name,payment.invoice_id,payment.bill_amount from customer natural join account natural join payment where payment.bill_amount >= 2000 and status = 'unpaid';

Output:

ebms/postgres@DBMS_Project

Query Editor | Query History

```
select customer.cust_id,customer.cust_name,payment.invoice_id,payment.bill_amount
from customer natural join account natural join payment
where payment.bill_amount >= 2000 and status = 'unpaid';
```

Data Output | Explain | Messages | Notifications

	cust_id integer	cust_name character varying (20)	invoice_id integer	bill_amount numeric (8,2)
1	101	Abdul Parmar	7001	2430.00
2	102	Shreyas Patel	8002	2126.00
3	103	Ashok Dinda	7003	2720.00
4	104	Justin Christian	8004	3799.00
5	111	Avneet Singh	8011	2805.00
6	113	Bhavya Chandrasala	8013	3279.00
7	115	Tanya Jagyasi	8015	2780.00
8	118	Vatsal Desai	8018	2904.00
9	120	Yuvraj Singh	8020	2668.00

6. Find the number of customers who have paid the bill using online transaction.

Query:

select count(c.cust_id) from customer as c inner join account as a on (c.cust_id = a.cust_id) inner join payment as p on (a.account_id = p.account_id) where payment_mode = 'online';

Output:

ebms/postgres@DBMS_Project

Query Editor | Query History

```
select count(c.cust_id) from customer as c inner join account as a on
(c.cust_id = a.cust_id) inner join payment as p on (a.account_id = p.account_id)
where payment_mode = 'online';
```

Data Output | Explain | Messages | Notifications

	count bigint
1	31

7. Total units consumed by all residents from Bandra in July 2021.

Query:

select sum(i.consumption_unit) from invoice as i inner join account as a on (i.account_id = a.account_id) inner join customer as c on (c.cust_id = a.cust_id) where c.area = 'Bandra' and i.reading_date >= '01-07-2021' and i.reading_date <= '31-07-2021';

Output:

ebms/postgres@DBMS_Project

Query Editor | Query History

```
select sum(i.consumption_unit) from invoice as i inner join account as a on
(i.account_id = a.account_id) inner join customer as c on (c.cust_id = a.cust_id)
where c.area = 'Bandra' and i.reading_date >= '01-07-2021'
and i.reading_date <= '31-07-2021'
```

Data Output | Explain | Messages | Notifications

	sum numeric
1	797.00

8. Retrieve number of customers who have not paid the bill in November 2021.

Query:

select count(i.invoice_id) from invoice as i inner join payment as p on (i.invoice_id = p.invoice_id) where i.due_date >= '01-11-2021' and i.due_date <= '30-11-2021' and p.status = 'unpaid';

Output:

ebms/postgres@DBMS_Project

Query Editor | Query History

```
select count(i.invoice_id) from invoice as i inner join payment as p on
(i.invoice_id = p.invoice_id) where i.due_date >= '01-11-2021'
and i.due_date <= '30-11-2021' and p.status = 'unpaid';
```

Data Output | Explain | Messages | Notifications

	count bigint
1	14

9. Show the feedback (feedback_id, feedback_date, type, cust_name) of given customer ID-102.

Query:

select c.cust_name,f.feedback_id,f.feedback_date,f.type from customer as c inner join account as a on(c.cust_id = a.cust_id) inner join feedback as f on (a.account_id = f.account_id) where c.cust_id = 102;

Output:

ebms/postgres@DBMS_Project

Query Editor | Query History

```
select c.cust_name,f.feedback_id,f.feedback_date,f.type from customer as c
inner join account as a on(c.cust_id = a.cust_id) inner join feedback as f
on (a.account_id = f.account_id) where c.cust_id = 102;
```

Data Output | Explain | Messages | Notifications

	cust_name character varying (20)	feedback_id integer	feedback_date date	type character varying (20)
1	Shreyas Patel	2021104	2021-07-18	Good
2	Shreyas Patel	2021105	2021-09-05	Excellent
3	Shreyas Patel	2021106	2021-11-04	Good

10. List {customer_name,cust_id,Address} for whom employee with emp_id = 390005 went for maintenance.

Query:

select c.cust_name,c.cust_id,c.address_line_1,c.area from customer as c inner join account as a on (c.cust_id = a.cust_id) inner join maintenance as m on (m.account_id = a.account_id) inner join employee as e on (e.emp_id = m.emp_id) where e.emp_id = 390005;

Output:

ebms/postgres@DBMS_Project

Query Editor | Query History

```
select c.cust_name,c.cust_id,c.address_line_1,c.area from customer as c
inner join account as a on (c.cust_id = a.cust_id) inner join maintenance as m
on (m.account_id = a.account_id) inner join employee as e on (e.emp_id = m.emp_id)
where e.emp_id = 390005;
```

Data Output | Explain | Messages | Notifications

	cust_name character varying (20)	cust_id [PK] integer	address_line_1 character varying (30)	area character varying (10)
1	Shreyas Patel	102	21, Gangutai Apartments	Bandra
2	Shreyas Patel	102	21, Gangutai Apartments	Bandra
3	Avneet Singh	111	12,Gangutai Apartments	Bandra
4	Avneet Singh	111	12,Gangutai Apartments	Bandra
5	Tanya Jagyasi	115	9, Gangutai Apartments	Bandra
6	Tanya Jagyasi	115	9, Gangutai Apartments	Bandra
7	Swati Singh	119	34, Mannat Bungalows	Bandra
8	Swati Singh	119	34, Mannat Bungalows	Bandra

11. List the cust_id and cust_name , area of all customers whose maintenance status are pending along with employee_name.

Query:

select c.cust_id,c.cust_name,c.area,m.maintenance_id,e.emp_name from customer as c inner join account as a on (a.cust_id = c.cust_id) inner join maintenance as m on (m.account_id = a.account_id) inner join employee as e on (e.emp_id = m.emp_id) where m.maintenance_status = 'pending';

Output:

ebms/postgres@DBMS_Project

Query Editor | Query History

```
select c.cust_id,c.cust_name,c.area,m.maintenance_id,e.emp_name from customer as c
inner join account as a on (a.cust_id = c.cust_id) inner join maintenance as m
on (m.account_id = a.account_id) inner join employee as e on (e.emp_id = m.emp_id)
where m.maintenance_status = 'pending';
```

Data Output | Explain | Messages | Notifications

	cust_id integer	cust_name character varying (20)	area character varying (10)	maintenance_id integer	emp_name character varying (50)
1	120	Yuvraj Singh	Andheri	9640	Hansraj Baldevraj
2	114	Pratik Patil	Andheri	9634	Hansraj Baldevraj
3	106	Jessica Jones	Andheri	9626	Hansraj Baldevraj
4	119	Swati Singh	Bandra	9639	Popatlal Bhatia
5	115	Tanya Jagyasi	Bandra	9635	Popatlal Bhatia
6	111	Avneet Singh	Bandra	9631	Popatlal Bhatia
7	105	Kalish Mehta	Borivali	9625	Ramubhai Patel
8	103	Ashok Dinda	Borivali	9623	Ramubhai Patel
9	109	Paul Matthew	Dahisar	9629	Champaklal Gada
10	104	Justin Christian	Dahisar	9624	Champaklal Gada

12. List the name, id and area of customer whose maintenance status are completed and feedback type is poor.

Query:

(select DISTINCT c.cust_id,c.cust_name,c.area from customer as c inner join account as a on (a.cust_id = c.cust_id) inner join maintenance as m on (m.account_id = a.account_id) where m.maintenance_status = 'completed') INTERSECT (select DISTINCT c.cust_id,c.cust_name,c.area from customer as c inner join account as a on (a.cust_id = c.cust_id) inner join feedback as f on (f.account_id = a.account_id) where f.type = 'Poor');

Output:

ebms/postgres@DBMS_Project

Query Editor | Query History

```
(select DISTINCT c.cust_id,c.cust_name,c.area from customer as c
 inner join account as a on (a.cust_id = c.cust_id) inner join maintenance as m
 on (m.account_id = a.account_id) where m.maintenance_status = 'completed')
 INTERSECT (select DISTINCT c.cust_id,c.cust_name,c.area from customer as c
 inner join account as a on (a.cust_id = c.cust_id)
 inner join feedback as f on (f.account_id = a.account_id)
 where f.type = 'Poor')
```

Data Output | Explain | Messages | Notifications

	cust_id integer	cust_name character varying (20)	area character varying (10)
1	113	Bhavya Chandrasala	Dahisar
2	115	Tanya Jagyasi	Bandra
3	105	Kalish Mehta	Borivali
4	116	Avi Kothari	Borivali
5	109	Paul Matthew	Dahisar

13. Find the customers and the account id and invoice no of the customers who have excess paid the paid amount.

Query:

select c.cust_id,c.cust_name,p.invoice_id,p.payment_date,p.excess_paid from customer as c inner join account as a on (a.cust_id = c.cust_id) inner join payment as p on (p.account_id = a.account_id) where p.excess_paid > 0;

Output:

ebms/postgres@DBMS_Project

Query Editor Query History

```
1 select c.cust_id,c.cust_name,p.invoice_id,p.payment_date,p.excess_paid
2 from customer as c inner join account as a on (a.cust_id = c.cust_id)
3 inner join payment as p on (p.account_id = a.account_id)
4 where p.excess_paid > 0;
```

Data Output Explain Messages Notifications

	cust_id integer	cust_name character varying (20)	invoice_id integer	payment_date date	excess_paid numeric (8,2)
1	103	Ashok Dinda	8003	2021-11-07	13.00
2	107	Luke Cage	8007	2021-11-05	20.00
3	108	Harisingh Mehta	7008	2021-09-20	5.00
4	110	Azzarhudin Gariwala	7010	2021-09-13	18.00
5	112	Shivani Rathi	8012	2021-11-09	62.00
6	112	Shivani Rathi	6012	2021-07-18	1.00
7	113	Bhavya Chandrasala	6013	2021-07-17	7.00
8	114	Pratik Patil	7014	2021-09-04	4.00
9	114	Pratik Patil	6014	2021-07-08	10.00
10	116	Avi Kothari	6016	2021-07-05	16.00
11	117	Jeffrey Joseph	8017	2021-11-09	2.00
12	117	Jeffrey Joseph	7017	2021-09-08	7.00
13	117	Jeffrey Joseph	6017	2021-07-18	25.00
14	118	Vatsal Desai	6018	2021-07-05	2.00
15	119	Swati Singh	6019	2021-07-23	2.00
16	120	Yuvraj Singh	6020	2021-07-12	1.00

14. Count the number of employees who are in Technical department.

Query:

select count(e.emp_id) from employee as e inner join department as d on (d.dept_no = e.dept_no) where d.dept_no = 2;

Output:

ebms/postgres@DBMS_Project

Query Editor | Query History

```
select count(e.emp_id) from employee as e inner join department as d
on (d.dept_no = e.dept_no) where d.dept_no = 2;
```

Data Output | Explain | Messages | Notifications

	count bigint
1	6

15. List all the customers (id, name, area, bill_amount) living in Bandra and borivali whose bill amount > 2000.

Query:

(select c.cust_name,c.cust_id,c.area,i.bill_amount from customer as c inner join account as a on (a.cust_id = c.cust_id) inner join invoice as i on (i.account_id = a.account_id) where i.bill_amount > 2000 and c.area = 'Bandra') UNION (select c.cust_name, c.cust_id,c.area,i.bill_amount from customer as c inner join account as a on (a.cust_id = c.cust_id) inner join invoice as i on (i.account_id = a.account_id) where i.bill_amount > 2000 and c.area = 'Borivali');

Output:

ebms/postgres@DBMS_Project

Query Editor | Query History

```
(select c.cust_name,c.cust_id,c.area,i.bill_amount from customer as c
 inner join account as a on (a.cust_id = c.cust_id) inner join invoice as i
 on (i.account_id = a.account_id) where i.bill_amount > 2000 and c.area = 'Bandra')
 UNION (select c.cust_name, c.cust_id,c.area,i.bill_amount from customer as c
 inner join account as a on (a.cust_id = c.cust_id) inner join invoice as i
 on (i.account_id = a.account_id) where i.bill_amount > 2000
 and c.area = 'Borivali');
```

Data Output | Explain | Messages | Notifications

	cust_name character varying (20)	cust_id integer	area character varying (10)	bill_amount numeric (8,2)
1	Ashok Dinda	103	Borivali	2016.60
2	Ashok Dinda	103	Borivali	2720.40
3	Avneet Singh	111	Bandra	2804.69
4	Azzarhudin Gariwala	110	Borivali	2031.90
5	Azzarhudin Gariwala	110	Borivali	2441.03
6	Azzarhudin Gariwala	110	Borivali	2445.00
7	Shivani Rathi	112	Borivali	2238.45
8	Shivani Rathi	112	Borivali	2643.90
9	Shreyas Patel	102	Bandra	2126.00
10	Tanya Jagyasi	115	Bandra	2033.80
11	Tanya Jagyasi	115	Bandra	2779.60
12	Vatsal Desai	118	Borivali	2904.00

16. **Get the details (acc_no, cust_name, area) of those customers whose maintenance was done by employee named 'Popatlal Bhatia'.**

Query:

select a.acc_no,c.cust_name,c.area from account as a inner join customer as c on (c.cust_id = a.cust_id) inner join maintenance as m on (a.account_id = m.account_id) inner join employee as e on (e.emp_id = m.emp_id) where emp_name = 'Popatlal Bhatia';

Output:

ebms/postgres@DBMS_Project

Query Editor | Query History

```
1 select a.acc_no,c.cust_name,c.area from account as a inner join
2 customer as c on (c.cust_id = a.cust_id) inner join maintenance
3 as m on (a.account_id = m.account_id) inner join employee as e
4 on (e.emp_id = m.emp_id) where emp_name = 'Popatlal Bhatia';
```

Data Output | Explain | Messages | Notifications

	acc_no bigint	cust_name character varying (20)	area character varying (10)
1	231485763	Shreyas Patel	Bandra
2	231485763	Shreyas Patel	Bandra
3	231485772	Avneet Singh	Bandra
4	231485772	Avneet Singh	Bandra
5	231485776	Tanya Jagyasi	Bandra
6	231485776	Tanya Jagyasi	Bandra
7	231485780	Swati Singh	Bandra
8	231485780	Swati Singh	Bandra

➢ Conclusion, Inference and Learning:

- We have tried to develop a system that can be a great help for the owner of any referred household i.e., the customer to receive bill from the electricity board.
  And also help the Electricity board to maintain records of customer bills.

- We have left all the options open so that if there is any other future requirement in the system by the admin or user for the enhancement of the system then it is possible to implement them.

- Though this project we got to know different entities involved in this system and how effortlessly this system works in real life. Customer details and his bills of previous transactions are displayed in a decent manner to his/her account. In this project we came to know that how the electricity company database stores the data and how the entities are linked to each other.

- Implementing this Database system on a large scale will be easy for the authorities as it can work on opensource database management systems like PostgreSQL and be effectively made available to the users for paying their electric bills and maintaining their account.

➢ Bibliography:

1. **PostgreSQL Tutorials**
   Link: https://postgresqltutorial.com/

2. **Maharashtra State Electricity Board (MSEB)**
   Link: http://www.msebindia.com/

3. **Link:** https://www.tutorialspoint.com/dbms/index.htm

4. **Link:** https://www.pgadmin.org/docs/pgadmin4/6.0/modifying_tables.html

5. **Link:** https://www.javatpoint.com/postgresql-tutorial