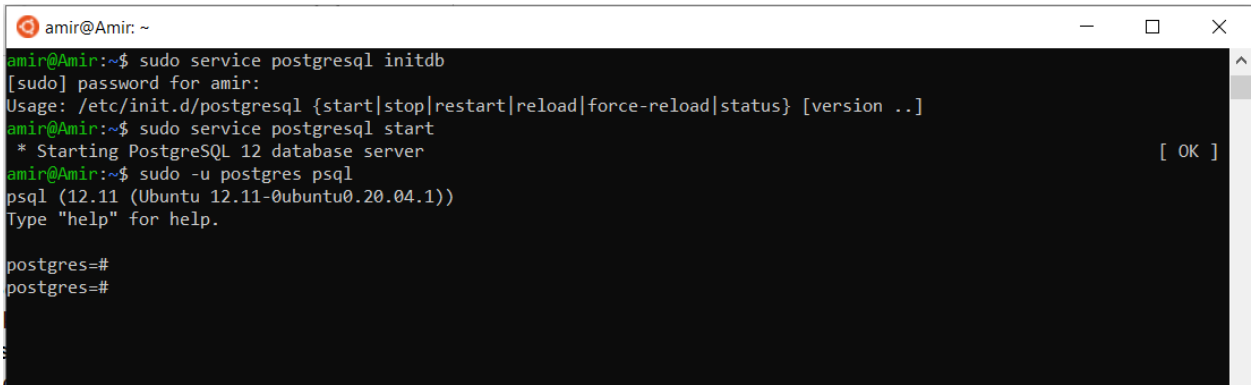
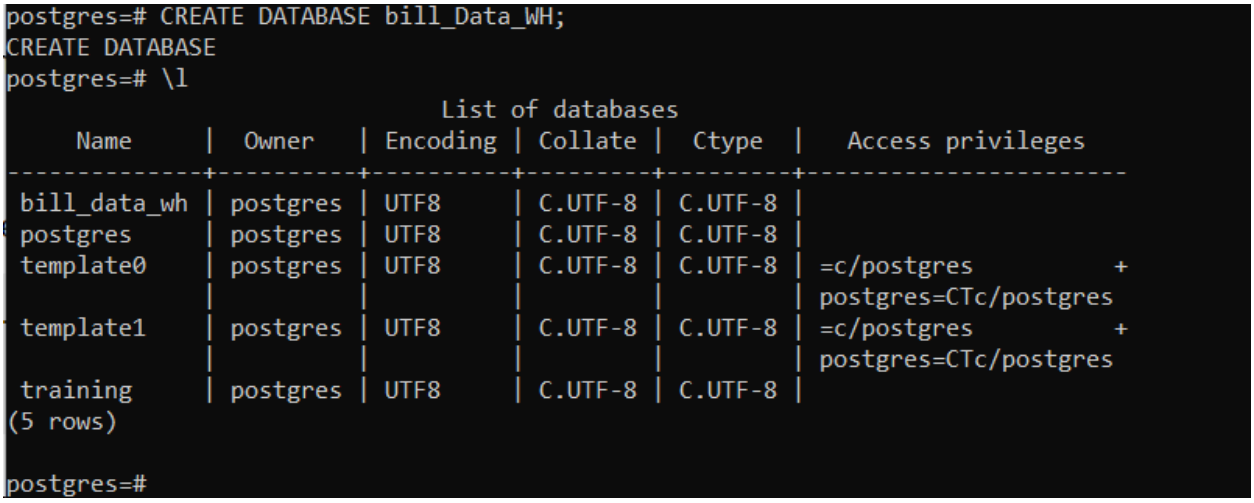
Let’s say that I have been assigned by a Senior Data Engineer to assist in creating a Staging server as part of the Data management workflow for an online startup store. The Senior Data Engineer has provided me with some steps to guide me in the creation of the staging area. He has also requested for screenshots showing the results of the steps that I have taken to arrive at the staging server that you have created.

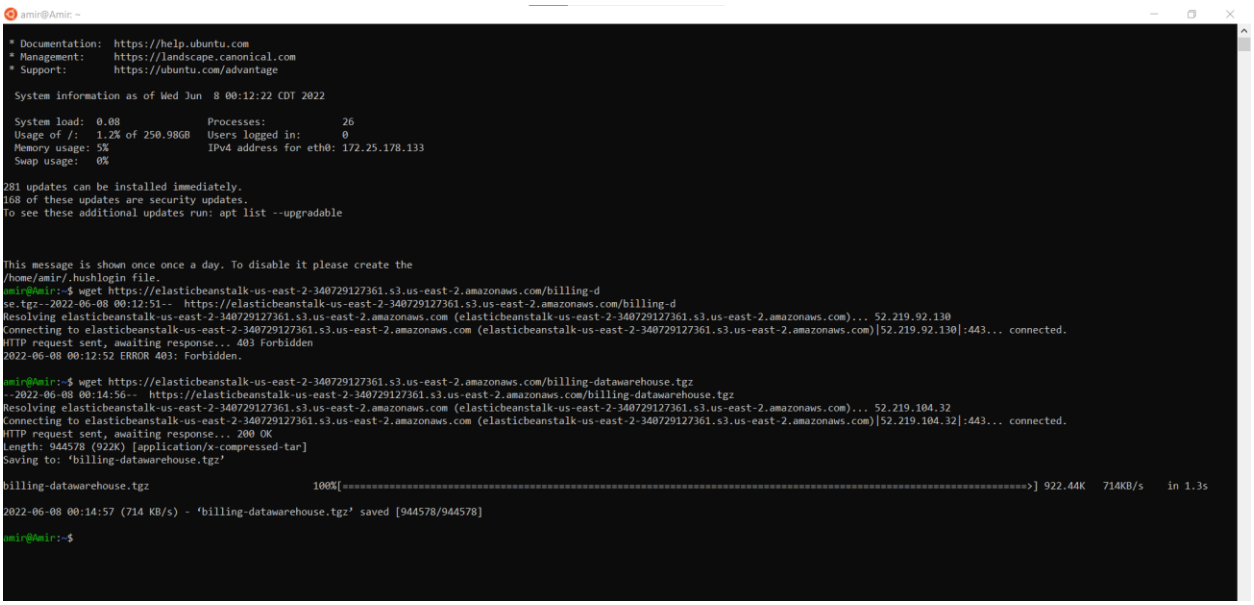
1 – Starting the database. For running the PostgreSQL service, there are 2 commands that we have to commit. Sudo service postgresql start & sudo -u postgres psql. After that, we will see the postgres=# in the command line, which means that we are in postgres environment, and we are using postgres user account.



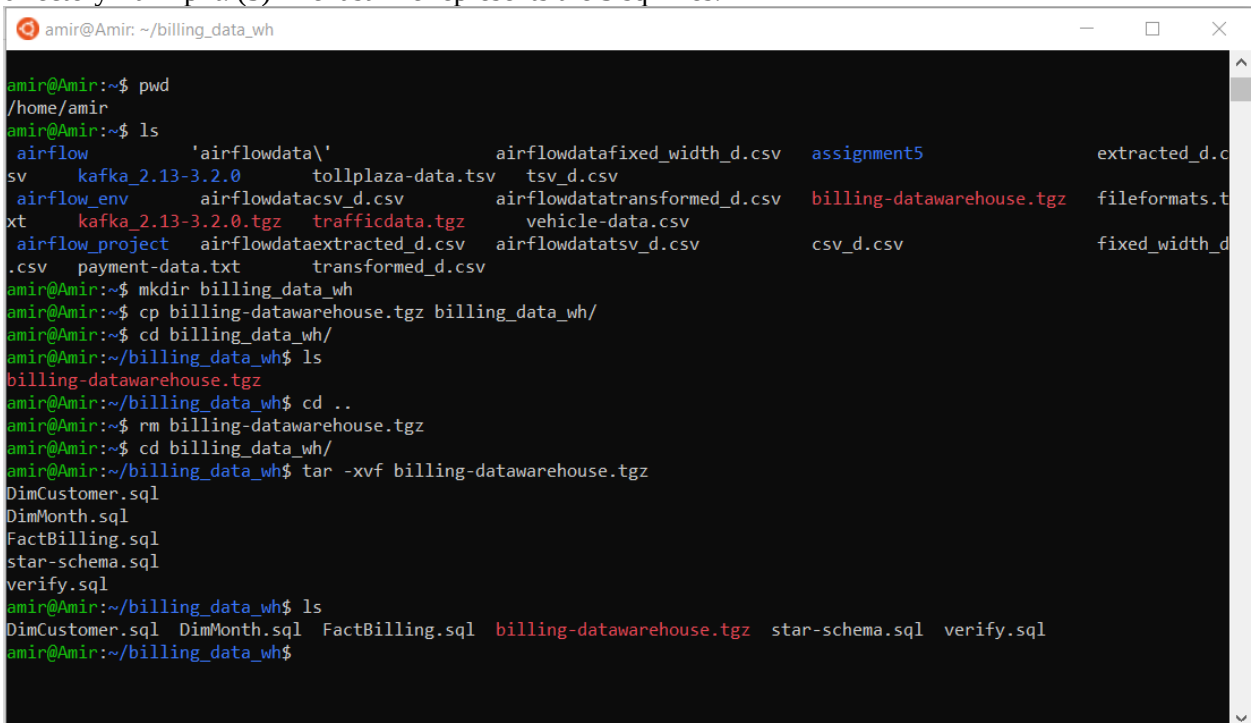
2- Create a database. The database name is bill_data_wh. \l command shows us the names of all the databases in the system. This way we can confirm the database creation.



3- Download the sql files for creating schemas and insertion intor the tables. For this step, I opened another linux command line (terminal) and use the wget command to download a rar file from it.



4 & 5 – (4) I have to unzip it. So using the tar -xvf command, I try to unzip it. But first I need to copy this file to a separate directory, in order not to make mess with the current directory. So, for this purpose, first I check the current directory, which is home. Then I list out the items in the home directory. Then I make a separate directory for the tgz file, and move it from home to its own directory. Then in the same directory I unzip it. (5) The last line represents the 5 sql files.



6 – Creating the Schema

6-1 – First table: FactBuilding. This table has 4 columns, one serial data type and three integers. But at the bigenning I have to connect to the database. \c bill_data_wh command establishes the connection.

```
postgres=# \c bill_data_wh;
You are now connected to database "bill_data_wh" as user "postgres".
bill_data_wh=# CREATE TABLE public."FactBilling"
(
billid serial,
customerid integer NOT NULL,
monthid integer NOT NULL,
billedamount integer NOT NULL,
PRIMARY KEY (billid)
);
CREATE TABLE
bill_data_wh=# \d
                  List of relations
 Schema |          Name          |   Type   |  Owner
--------+------------------------+----------+----------
 public | FactBilling            | table    | postgres
 public | FactBilling_billid_seq | sequence | postgres
(2 rows)

bill_data_wh=#
```

6-2 – Second table: DimMonth. This table contains all the information of the month related to the monthid column of the FactBilling

```
bill_data_wh=# CREATE TABLE public."DimMonth"
(
monthid integer NOT NULL,
year integer NOT NULL,
month integer NOT NULL,
monthname varchar(10) NOT NULL,
quarter integer NOT NULL,
quartername varchar(2) NOT NULL,
PRIMARY KEY (monthid)
);
CREATE TABLE
bill_data_wh=# \d
                  List of relations
 Schema |          Name          |   Type   |  Owner
--------+------------------------+----------+----------
 public | DimMonth               | table    | postgres
 public | FactBilling            | table    | postgres
 public | FactBilling_billid_seq | sequence | postgres
(3 rows)

bill_data_wh=#
```

6-3 – Third table: DimCustomer. Likewise the previous table, this table holds all the information regarding the customerid column of the FactBilling table.

```
bill_data_wh=# CREATE TABLE public."DimCustomer"
(
customerid integer NOT NULL,
category varchar(10) NOT NULL,
country varchar(40) NOT NULL,
industry varchar(40) NOT NULL,
PRIMARY KEY (customerid)
);
CREATE TABLE
bill_data_wh=# \d
                 List of relations
 Schema |          Name          |   Type   |  Owner
--------+------------------------+----------+----------
 public | DimCustomer            | table    | postgres
 public | DimMonth               | table    | postgres
 public | FactBilling            | table    | postgres
 public | FactBilling_billid_seq | sequence | postgres
(4 rows)

bill_data_wh=#
```

6-4 – Altering the FactBilling table. In this step, I add a foreign key to the table. Foreign key is the key to relate to another tables' primary key. This way the relational databases work. So in this picture, I make two foreign keys in factbilling, the first one is customerid, which is going to reference the same customerid in DimMonth table, and the second one is monthid, and it is going to be referred by the monthid of the DimMonth table.

```
bill_data_wh=# ALTER TABLE public."FactBilling"
bill_data_wh-# ADD FOREIGN KEY (customerid)
bill_data_wh-# REFERENCES public."DimCustomer" (customerid)
bill_data_wh-# NOT VALID;
ALTER TABLE
bill_data_wh=# ALTER TABLE public."FactBilling"
bill_data_wh-# ADD FOREIGN KEY (monthid)
bill_data_wh-# REFERENCES public."DimMonth" (monthid)
bill_data_wh-# NOT VALID;
ALTER TABLE
bill_data_wh=#
```

7 – Loading the dimension into the DimCustomer table. As there are roughly 1000 rows of data to enter into the table, it is good to have a command that reads the ".sql" file, and inserts the pre-written rows into the table. So for this purpose, the command is \i 'destination_of_the_sql_file'. I use this command to insert into the table.

```
amir@Amir: ~
bill_data_wh=#
bill_data_wh=# \i '/home/amir/billing_data_wh/DimCustomer.sql'
INSERT 0 1000
bill_data_wh=#
```

BTW, next picture is the .sql file that is already inserted (just 23 row od 1000 data rows):

```sql
INSERT INTO "DimCustomer"(customerid,category,country,industry)
VALUES
(1,'Individual','Indonesia','Engineering'),
(614,'Individual','United States','Product Management'),
(615,'Individual','China','Services'),
(616,'Individual','Russia','Accounting'),
(617,'Individual','Chile','Business Development'),
(618,'Individual','Nicaragua','Human Resources'),
(41,'Company','Brazil','Marketing'),
(619,'Individual','Russia','Business Development'),
(620,'Individual','China','Business Development'),
(956,'Individual','Peru','Research and Development'),
(621,'Individual','Angola','Services'),
(622,'Individual','Poland','Legal'),
(623,'Individual','Italy','Training'),
(624,'Individual','Indonesia','Sales'),
(625,'Individual','Portugal','Services'),
(626,'Individual','Mexico','Engineering'),
(40,'Individual','Portugal','Human Resources'),
(627,'Company','Philippines','Services'),
(628,'Individual','France','Business Development'),
(629,'Company','France','Training'),
(630,'Individual','Indonesia','Services'),
(631,'Company','Poland','Marketing'),
(632,'Company','Brunei','Training'),
```

8 – The same process for the DimMonth table.

```
bill_data_wh=# \i '/home/amir/billing_data_wh/DimMonth.sql'
INSERT 0 132
bill_data_wh=#
```

The next picture shows some data rows of the sql file.

```sql
INSERT INTO "DimMonth"
(monthid,year,month,monthname,quarter,quartername)
VALUES
(20091, 2009, 1, 'Janauary', 1, 'Q1'),
(200910, 2009, 10, 'October', 4, 'Q4'),
(200911, 2009, 11, 'November', 4, 'Q4'),
(200912, 2009, 12, 'December', 4, 'Q4'),
(20092, 2009, 2, 'February', 1, 'Q1'),
(20093, 2009, 3, 'March', 1, 'Q1'),
(20094, 2009, 4, 'April', 2, 'Q2'),
(20095, 2009, 5, 'May', 2, 'Q2'),
(20096, 2009, 6, 'June', 2, 'Q2'),
(20097, 2009, 7, 'July', 3, 'Q3'),
(20098, 2009, 8, 'August', 3, 'Q3'),
(20099, 2009, 9, 'September', 3, 'Q3'),
(20101, 2010, 1, 'Janauary', 1, 'Q1'),
(201010, 2010, 10, 'October', 4, 'Q4'),
(201011, 2010, 11, 'November', 4, 'Q4'),
(201012, 2010, 12, 'December', 4, 'Q4'),
(20102, 2010, 2, 'February', 1, 'Q1'),
(20103, 2010, 3, 'March', 1, 'Q1'),
(20104, 2010, 4, 'April', 2, 'Q2'),
(20105, 2010, 5, 'May', 2, 'Q2'),
(20106, 2010, 6, 'June', 2, 'Q2'),
(20107, 2010, 7, 'July', 3, 'Q3'),
```

9 – The same step for FactBilling table happens. I have to mention the this third table took to long to be inserted. 132000 of data rows, it is huge.

```
amir@Amir: ~
bill_data_wh=# \i '/home/amir/billing_data_wh/DimCustomer.sql'
INSERT 0 1000
bill_data_wh=# \i '/home/amir/billing_data_wh/DimMonth.sql'
INSERT 0 132
bill_data_wh=# \i '/home/amir/billing_data_wh/FactBilling.sql'
INSERT 0 132000
bill_data_wh=#
```

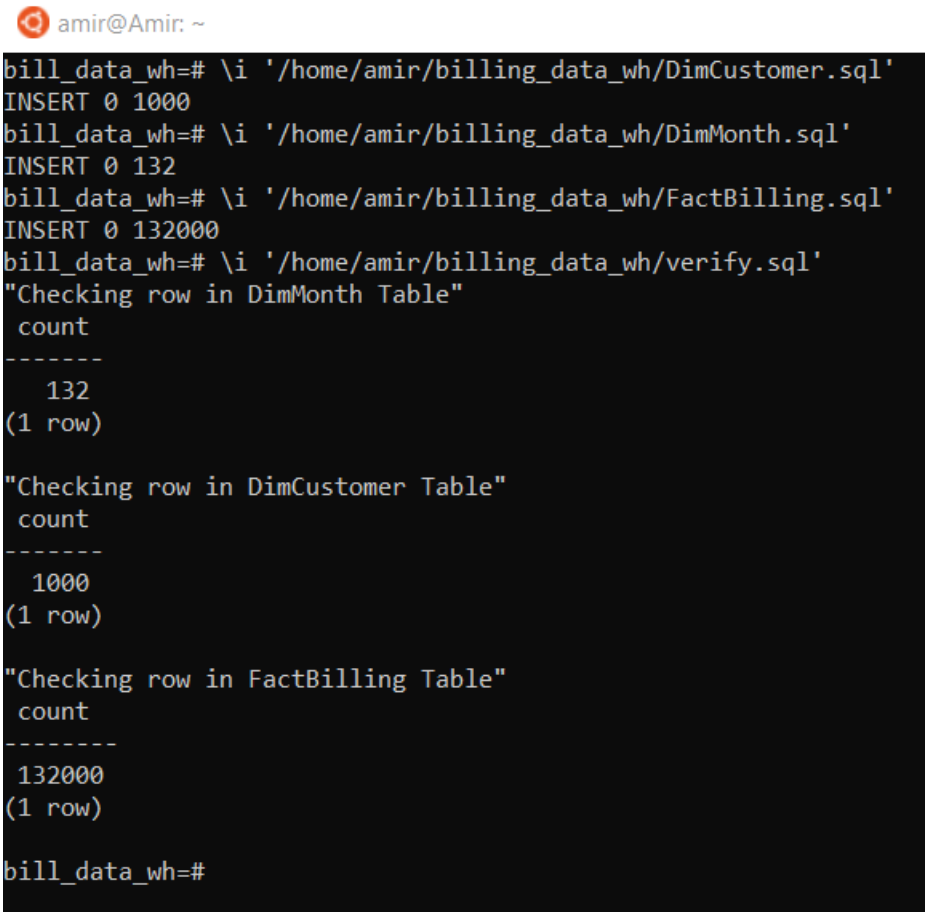
Again, just a glance to the sql file:

```
INSERT INTO "FactBilling"
(billid,customerid,billedamount,monthid)
VALUES
(1,1,5060,20091),
(2,614,9638,20091),
(3,615,11573,20091),
(4,616,18697,20091),
(5,617,944,20091),
(6,618,3539,20091),
(7,41,6591,20091),
(8,619,16061,20091),
(9,620,1250,20091),
(10,956,15105,20091),
```

10 – Verifying the Data Insertion. In the next picture I represent the sql file which shows the command for requesting the count of data for each table.

```
\echo "Checking row in DimMonth Table"
select count(*) from "DimMonth";
\echo "Checking row in DimCustomer Table"
select count(*) from "DimCustomer";
\echo "Checking row in FactBilling Table"
select count(*) from "FactBilling";
```

And by running it the same way as insertion process, I will have this output:



So this shows that all of the data have ben inserted into their tables, safe and sound.