DBMS PROJECT REPORT - ASSIGNMENT 4

COURIER MANAGEMENT SYSTEM

Team no 4:

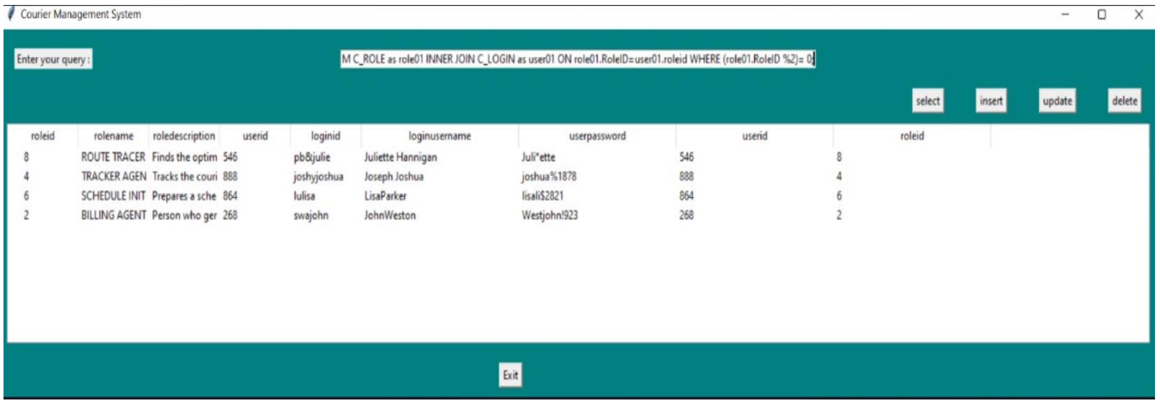
NAME:	SRN:
Ruchita V R	PES1UG19CS397
Rajeshwari R	PES1UG19CS375
Ramya C	PES1UG19CS379

Dependencies installed for the database connectivity

Packages installed	Modules imported
Tkinter, Pandas, pandastable, Psycopg2	Ttk, StringVar from tkinter, Table, TableModel from pandastable,

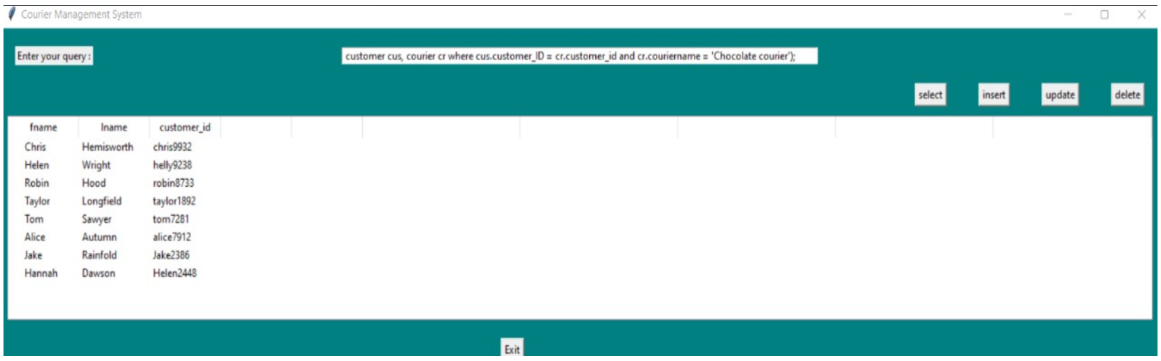
Tkinter module is used for creating the GUI for the frontend. The tk toolkit is used to construct the widgets like buttons, data fields, etc. StringVar is used to store names as a string. Pandas library is used for data representation and data flexibility. Table and TableModel module from the pandastable package is used to create a frame to place the table and access the table elements. Psycopg2 is imported as the postgresql database adapter for python.

Screenshots of the statement executed from the front end



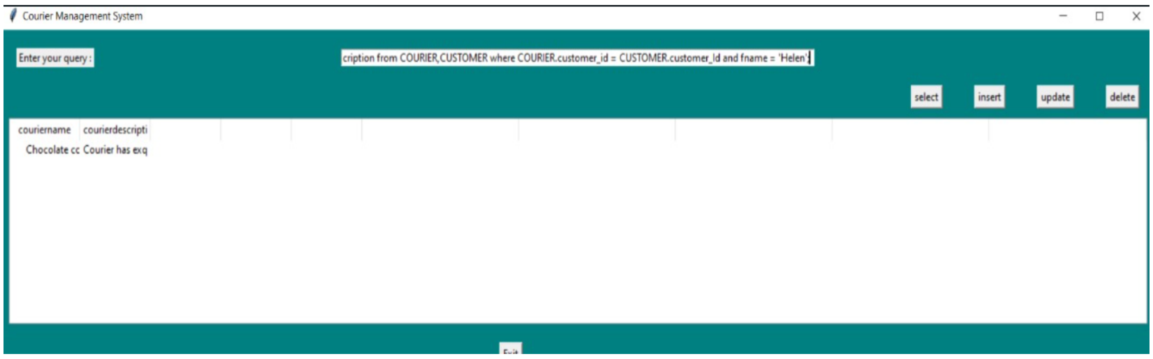
```
M C_ROLE as role01 INNER JOIN C_LOGIN as user01 ON role01.RoleID=user01.roleid WHERE (role01.RoleID %2)= 0
```

roleid	rolename	roledescription	userid	loginid	loginusername	userpassword	userid	roleid
8	ROUTE TRACER	Finds the optim	546	pb&julie	Juliette Hannigan	Juli*ette	546	8
4	TRACKER AGEN	Tracks the couri	888	joshyjoshua	Joseph Joshua	joshua%1878	888	4
6	SCHEDULE INIT	Prepares a sche	864	lulisa	LisaParker	lisali$2821	864	6
2	BILLING AGENT	Person who ger	268	swajohn	JohnWeston	Westjohn!923	268	2



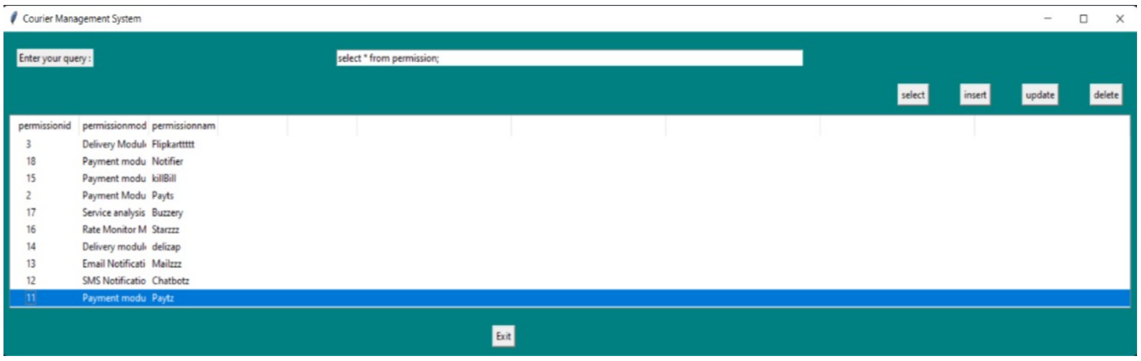
```
customer cus, courier cr where cus.customer_ID = cr.customer_id and cr.couriername = 'Chocolate courier');
```

fname	lname	customer_id
Chris	Hemisworth	chris9932
Helen	Wright	helly9238
Robin	Hood	robin8733
Taylor	Longfield	taylor1892
Tom	Sawyer	tom7281
Alice	Autumn	alice7912
Jake	Rainfold	Jake2386
Hannah	Dawson	Helen2448



```
cription from COURIER,CUSTOMER where COURIER.customer_id = CUSTOMER.customer_Id and fname = 'Helen';
```

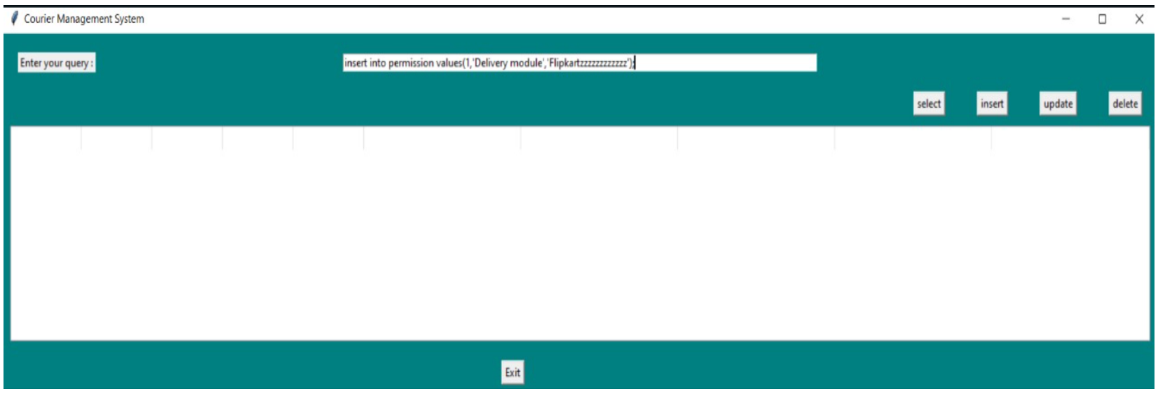
couriername	courierdescripti
Chocolate cc	Courier has exq



```
select * from permission;
```

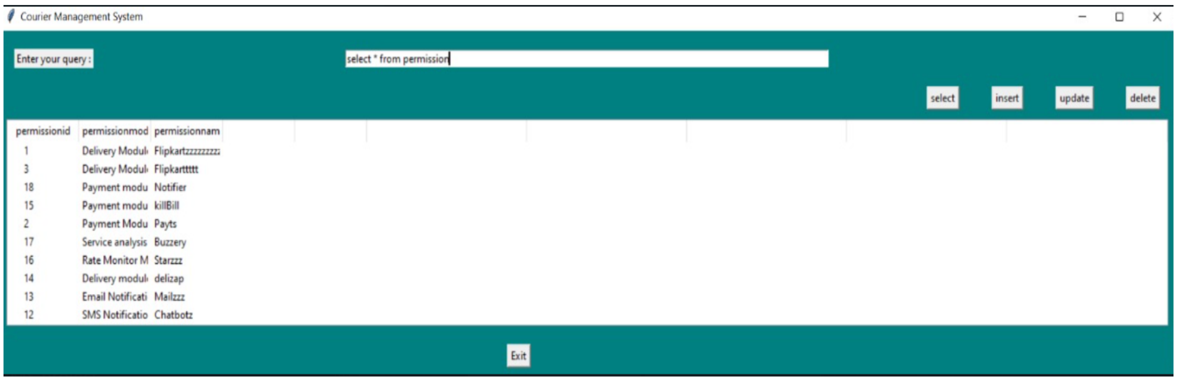
permissionid	permissionmod	permissionnam
3	Delivery Modul	Flipkarttttt
18	Payment modu	Notifier
15	Payment modu	killBill
2	Payment Modu	Payts
17	Service analysis	Buzzery
16	Rate Monitor M	Starzzz
14	Delivery modul	delizap
13	Email Notificati	Mailzzz
12	SMS Notificatio	Chatbotz
11	Payment modu	Paytz

Inserting values into the table,



```
insert into permission values(1,'Delivery module','Flipkartzzzzzzzzzzzzz');
```

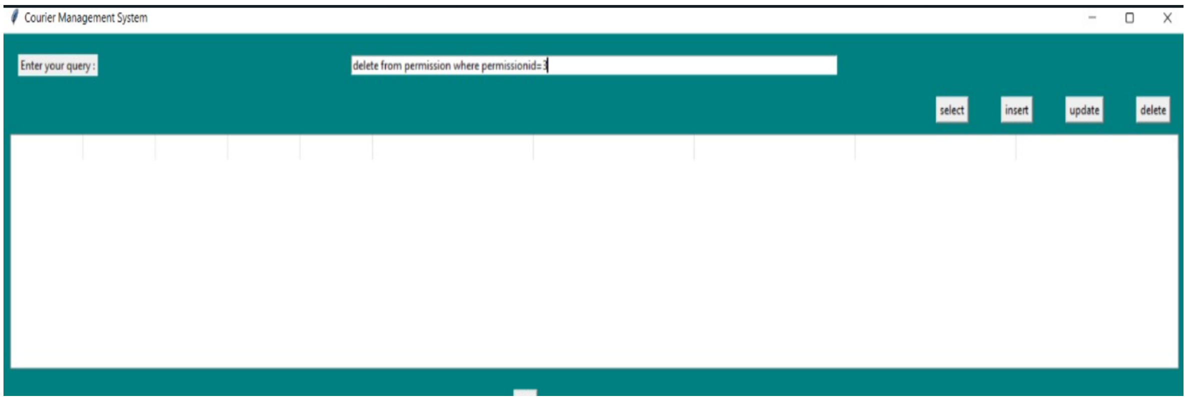
After insertion,



```
select * from permission
```

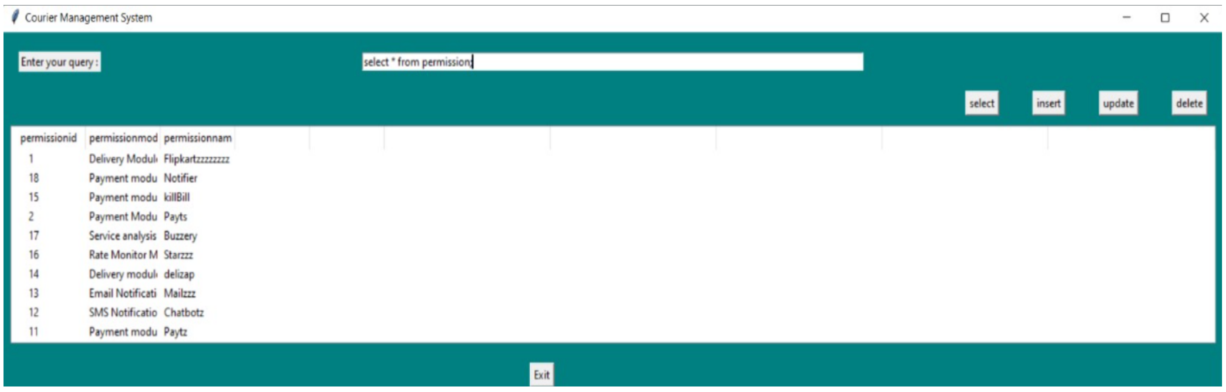
permissionid	permissionmod	permissionnam
1	Delivery Modul	Flipkartzzzzzzzz
3	Delivery Modul	Flipkartttttt
18	Payment modu	Notifier
15	Payment modu	killBill
2	Payment Modu	Payts
17	Service analysis	Buzzery
16	Rate Monitor M	Starzzz
14	Delivery modul	delizap
13	Email Notificati	Mailzzz
12	SMS Notificatio	Chatbotz

Deleting values from the table,



```
delete from permission where permissionid=3
```

After deletion,



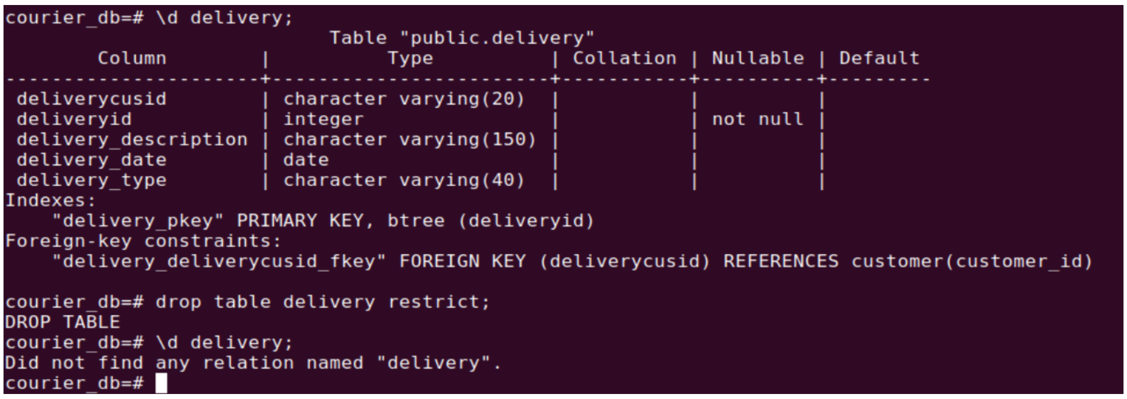
```
select * from permission;
```

permissionid	permissionmod	permissionnam
1	Delivery Modul	Flipkartzzzzzzzz
18	Payment modu	Notifier
15	Payment modu	killBill
2	Payment Modu	Payts
17	Service analysis	Buzzery
16	Rate Monitor M	Starzzz
14	Delivery modul	delizap
13	Email Notificati	Mailzzz
12	SMS Notificatio	Chatbotz
11	Payment modu	Paytz

<u>***Screenshots for the schema change statements***</u>

Dropping the schema using restrict,

```
DROP TABLE delivery RESTRICT;
```

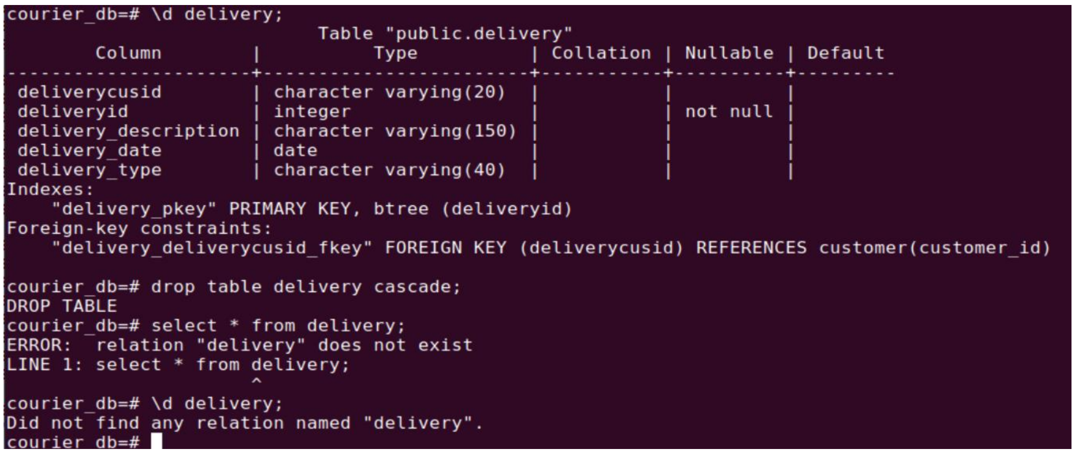


```
courier_db=# \d delivery;
                              Table "public.delivery"
       Column         |          Type          | Collation | Nullable | Default
----------------------+------------------------+-----------+----------+---------
 deliverycusid        | character varying(20)  |           |          |
 deliveryid           | integer                |           | not null |
 delivery_description | character varying(150) |           |          |
 delivery_date        | date                   |           |          |
 delivery_type        | character varying(40)  |           |          |
Indexes:
    "delivery_pkey" PRIMARY KEY, btree (deliveryid)
Foreign-key constraints:
    "delivery_deliverycusid_fkey" FOREIGN KEY (deliverycusid) REFERENCES customer(customer_id)

courier_db=# drop table delivery restrict;
DROP TABLE
courier_db=# \d delivery;
Did not find any relation named "delivery".
courier_db=#
```

Dropping the schema using cascade,

DROP TABLE delivery CASCADE;



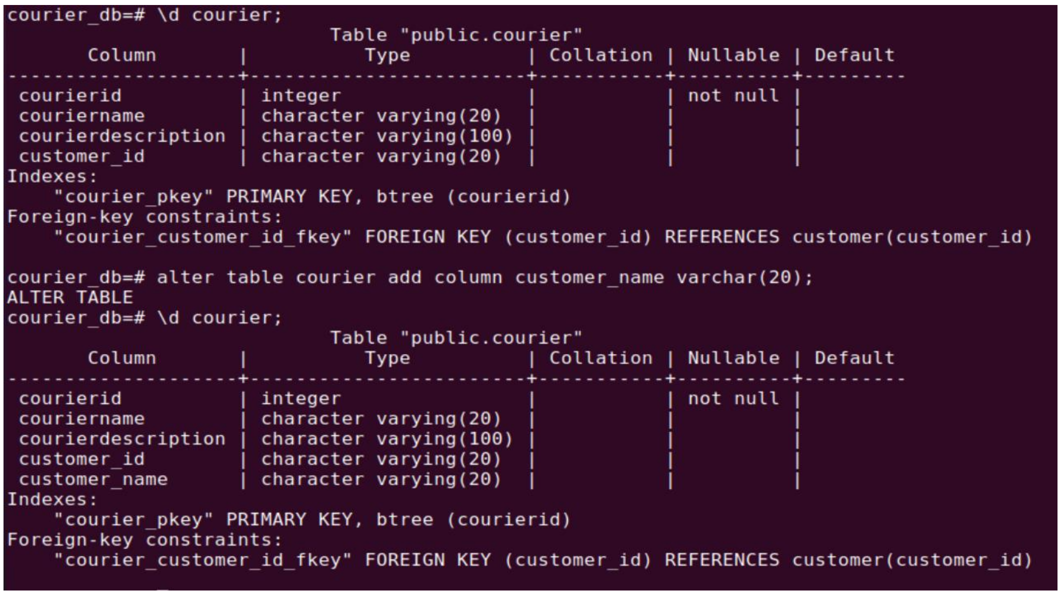
```
courier_db=# \d delivery;
                              Table "public.delivery"
       Column         |          Type          | Collation | Nullable | Default
----------------------+------------------------+-----------+----------+---------
 deliverycusid        | character varying(20)  |           |          |
 deliveryid           | integer                |           | not null |
 delivery_description | character varying(150) |           |          |
 delivery_date        | date                   |           |          |
 delivery_type        | character varying(40)  |           |          |
Indexes:
    "delivery_pkey" PRIMARY KEY, btree (deliveryid)
Foreign-key constraints:
    "delivery_deliverycusid_fkey" FOREIGN KEY (deliverycusid) REFERENCES customer(customer_id)

courier_db=# drop table delivery cascade;
DROP TABLE
courier_db=# select * from delivery;
ERROR:  relation "delivery" does not exist
LINE 1: select * from delivery;
                      ^
courier_db=# \d delivery;
Did not find any relation named "delivery".
courier_db=#
```

Adding a new column to a schema,

A new column customer_name has been added to the existing schema as shown below, using the command

ALTER TABLE courier ADD COLUMN customer_name varchar(20);



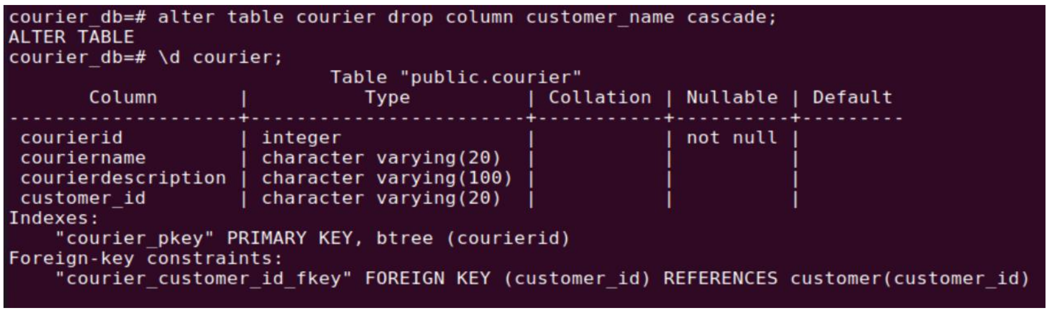
```
courier_db=# \d courier;
                              Table "public.courier"
       Column        |          Type          | Collation | Nullable | Default
---------------------+------------------------+-----------+----------+---------
 courierid           | integer                |           | not null |
 couriername         | character varying(20)  |           |          |
 courierdescription  | character varying(100) |           |          |
 customer_id         | character varying(20)  |           |          |
Indexes:
    "courier_pkey" PRIMARY KEY, btree (courierid)
Foreign-key constraints:
    "courier_customer_id_fkey" FOREIGN KEY (customer_id) REFERENCES customer(customer_id)

courier_db=# alter table courier add column customer_name varchar(20);
ALTER TABLE
courier_db=# \d courier;
                              Table "public.courier"
       Column        |          Type          | Collation | Nullable | Default
---------------------+------------------------+-----------+----------+---------
 courierid           | integer                |           | not null |
 couriername         | character varying(20)  |           |          |
 courierdescription  | character varying(100) |           |          |
 customer_id         | character varying(20)  |           |          |
 customer_name       | character varying(20)  |           |          |
Indexes:
    "courier_pkey" PRIMARY KEY, btree (courierid)
Foreign-key constraints:
    "courier_customer_id_fkey" FOREIGN KEY (customer_id) REFERENCES customer(customer_id)
```

Deleting a column from a schema,

A column is dropped from the schema using the drop column.

ALTER TABLE courier DROP COLUMN customer_name CASCADE

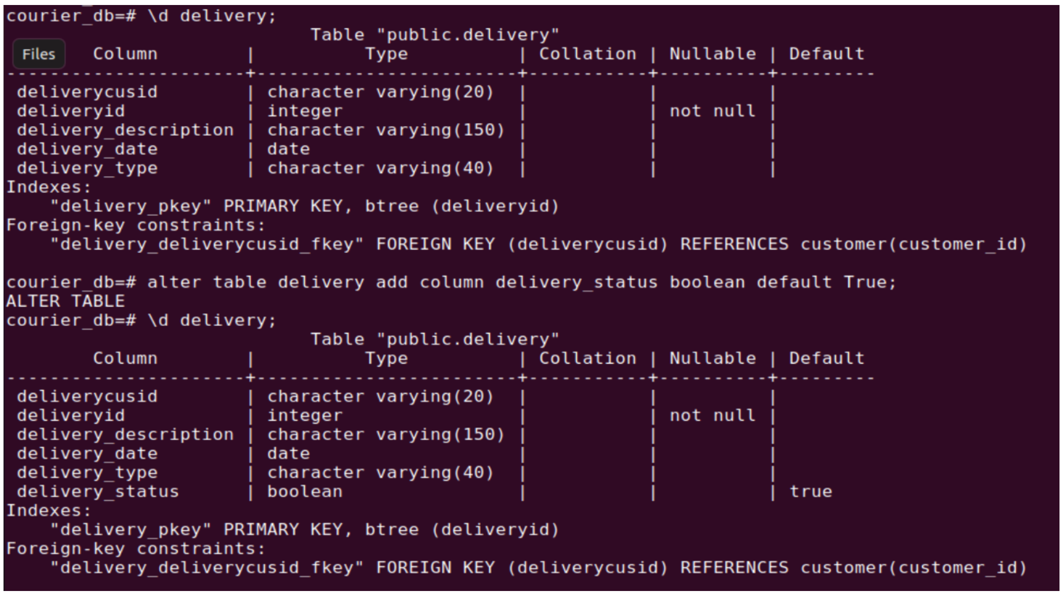


```
courier_db=# alter table courier drop column customer_name cascade;
ALTER TABLE
courier_db=# \d courier;
                          Table "public.courier"
       Column       |          Type          | Collation | Nullable | Default
--------------------+------------------------+-----------+----------+---------
 courierid          | integer                |           | not null |
 couriername        | character varying(20)  |           |          |
 courierdescription | character varying(100) |           |          |
 customer_id        | character varying(20)  |           |          |
Indexes:
    "courier_pkey" PRIMARY KEY, btree (courierid)
Foreign-key constraints:
    "courier_customer_id_fkey" FOREIGN KEY (customer_id) REFERENCES customer(customer_id)
```

Adding a column with the default value,

Here we add a column delivery_status which tells if the courier is delivered or not and by default, it is set to true meaning it is already delivered.

ALTER TABLE delivery ADD COLUMN delivery_status Boolean DEFAULT True;



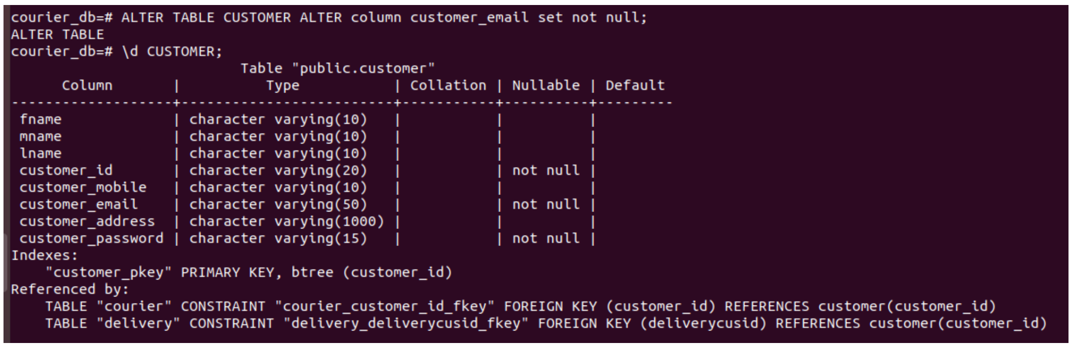
```
courier_db=# \d delivery;
                          Table "public.delivery"
       Column        |          Type          | Collation | Nullable | Default
---------------------+------------------------+-----------+----------+---------
 deliverycusid       | character varying(20)  |           |          |
 deliveryid          | integer                |           | not null |
 delivery_description | character varying(150) |           |          |
 delivery_date       | date                   |           |          |
 delivery_type       | character varying(40)  |           |          |
Indexes:
    "delivery_pkey" PRIMARY KEY, btree (deliveryid)
Foreign-key constraints:
    "delivery_deliverycusid_fkey" FOREIGN KEY (deliverycusid) REFERENCES customer(customer_id)

courier_db=# alter table delivery add column delivery_status boolean default True;
ALTER TABLE
courier_db=# \d delivery;
                          Table "public.delivery"
       Column        |          Type          | Collation | Nullable | Default
---------------------+------------------------+-----------+----------+---------
 deliverycusid       | character varying(20)  |           |          |
 deliveryid          | integer                |           | not null |
 delivery_description | character varying(150) |           |          |
 delivery_date       | date                   |           |          |
 delivery_type       | character varying(40)  |           |          |
 delivery_status     | boolean                |           |          | true
Indexes:
    "delivery_pkey" PRIMARY KEY, btree (deliveryid)
Foreign-key constraints:
    "delivery_deliverycusid_fkey" FOREIGN KEY (deliverycusid) REFERENCES customer(customer_id)
```

<u>Screenshots of the constraint changes,</u>

1. Add Not null contraint on phone number in customer table

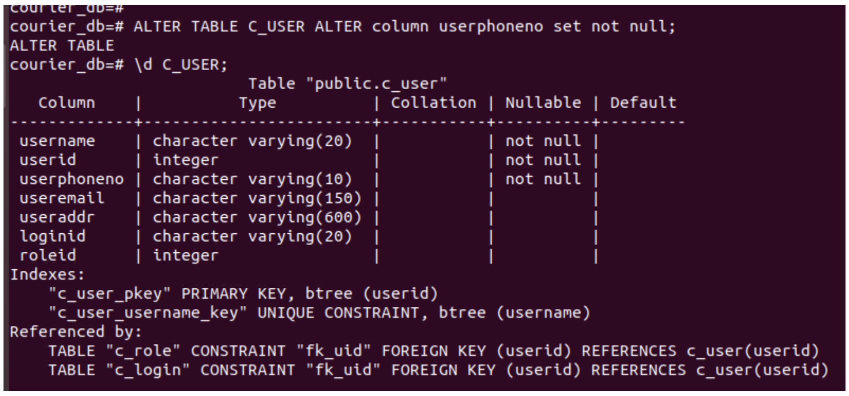
ALTER TABLE CUSTOMER ALTER column customer_email set not null;



```
courier_db=# ALTER TABLE CUSTOMER ALTER column customer_email set not null;
ALTER TABLE
courier_db=# \d CUSTOMER;
                        Table "public.customer"
      Column      |          Type           | Collation | Nullable | Default
------------------+-------------------------+-----------+----------+---------
 fname            | character varying(10)   |           |          |
 mname            | character varying(10)   |           |          |
 lname            | character varying(10)   |           |          |
 customer_id      | character varying(20)   |           | not null |
 customer_mobile  | character varying(10)   |           |          |
 customer_email   | character varying(50)   |           | not null |
 customer_address | character varying(1000) |           |          |
 customer_password| character varying(15)   |           | not null |
Indexes:
    "customer_pkey" PRIMARY KEY, btree (customer_id)
Referenced by:
    TABLE "courier" CONSTRAINT "courier_customer_id_fkey" FOREIGN KEY (customer_id) REFERENCES customer(customer_id)
    TABLE "delivery" CONSTRAINT "delivery_deliverycusid_fkey" FOREIGN KEY (deliverycusid) REFERENCES customer(customer_id)
```

2. Add not null constraint on phone number in c_user table

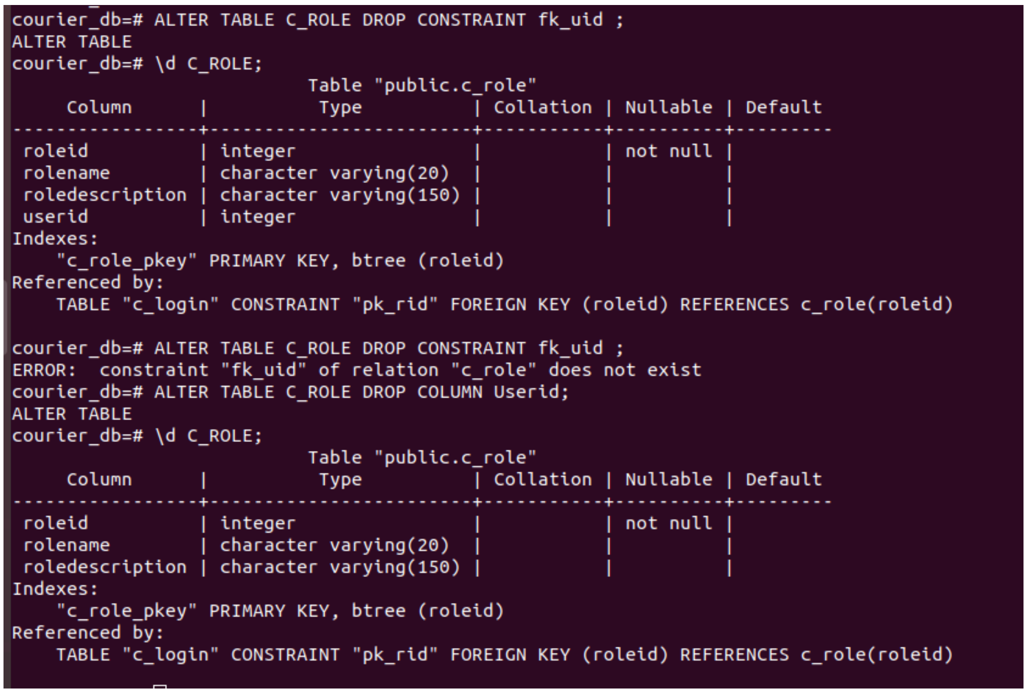
   ALTER TABLE C_USER ALTER column userphoneno set not null;



```
courier_db=#
courier_db=# ALTER TABLE C_USER ALTER column userphoneno set not null;
ALTER TABLE
courier_db=# \d C_USER;
                     Table "public.c_user"
   Column    |          Type          | Collation | Nullable | Default
-------------+------------------------+-----------+----------+---------
 username    | character varying(20)  |           | not null |
 userid      | integer                |           | not null |
 userphoneno | character varying(10)  |           | not null |
 useremail   | character varying(150) |           |          |
 useraddr    | character varying(600) |           |          |
 loginid     | character varying(20)  |           |          |
 roleid      | integer                |           |          |
Indexes:
    "c_user_pkey" PRIMARY KEY, btree (userid)
    "c_user_username_key" UNIQUE CONSTRAINT, btree (username)
Referenced by:
    TABLE "c_role" CONSTRAINT "fk_uid" FOREIGN KEY (userid) REFERENCES c_user(userid)
    TABLE "c_login" CONSTRAINT "fk_uid" FOREIGN KEY (userid) REFERENCES c_user(userid)
```

3. Remove userid from role table

   ALTER TABLE C_ROLE DROP CONSTRAINT fk_uid ;
   ALTER TABLE C_ROLE DROP COLUMN Userid;

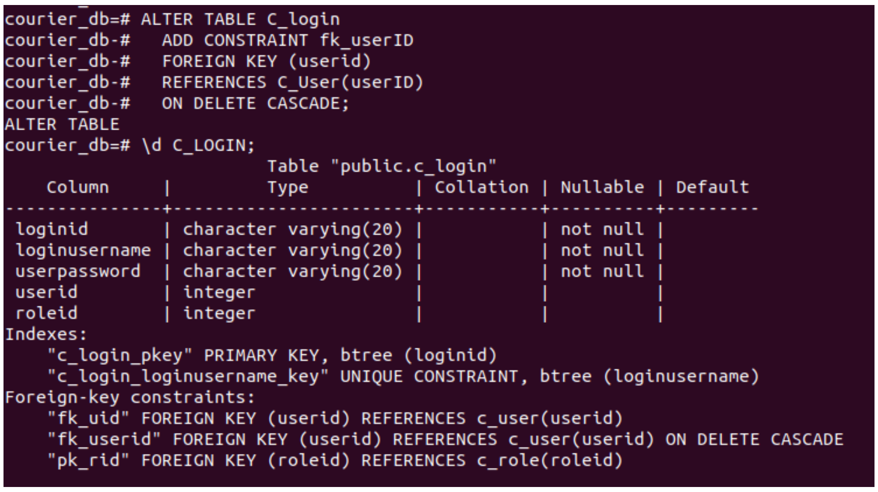


```
courier_db=# ALTER TABLE C_ROLE DROP CONSTRAINT fk_uid ;
ALTER TABLE
courier_db=# \d C_ROLE;
                        Table "public.c_role"
     Column      |          Type          | Collation | Nullable | Default
-----------------+------------------------+-----------+----------+---------
 roleid          | integer                |           | not null |
 rolename        | character varying(20)  |           |          |
 roledescription | character varying(150) |           |          |
 userid          | integer                |           |          |
Indexes:
    "c_role_pkey" PRIMARY KEY, btree (roleid)
Referenced by:
    TABLE "c_login" CONSTRAINT "pk_rid" FOREIGN KEY (roleid) REFERENCES c_role(roleid)

courier_db=# ALTER TABLE C_ROLE DROP CONSTRAINT fk_uid ;
ERROR:  constraint "fk_uid" of relation "c_role" does not exist
courier_db=# ALTER TABLE C_ROLE DROP COLUMN Userid;
ALTER TABLE
courier_db=# \d C_ROLE;
                        Table "public.c_role"
     Column      |          Type          | Collation | Nullable | Default
-----------------+------------------------+-----------+----------+---------
 roleid          | integer                |           | not null |
 rolename        | character varying(20)  |           |          |
 roledescription | character varying(150) |           |          |
Indexes:
    "c_role_pkey" PRIMARY KEY, btree (roleid)
Referenced by:
    TABLE "c_login" CONSTRAINT "pk_rid" FOREIGN KEY (roleid) REFERENCES c_role(roleid)
```

4. Add on delete cascade on delete command on user table - loginid and password must be removed from login table

    ALTER TABLE C_login ADD CONSTRAINT fk_userID FOREIGN KEY (userid) REFERENCES C_User(userID) ON DELETE CASCADE;

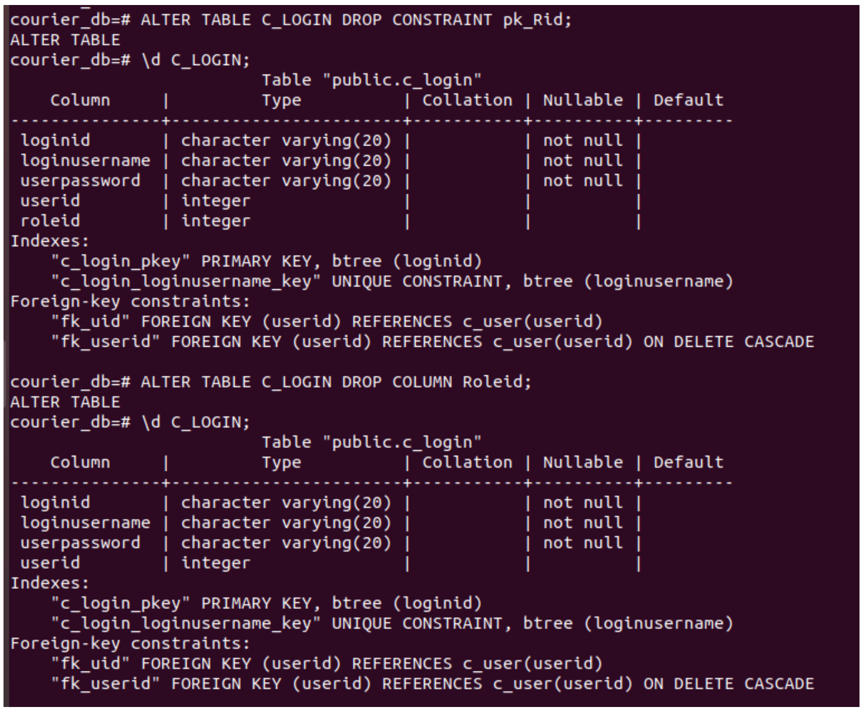


```
courier_db=# ALTER TABLE C_login
courier_db-#   ADD CONSTRAINT fk_userID
courier_db-#   FOREIGN KEY (userid)
courier_db-#   REFERENCES C_User(userID)
courier_db-#   ON DELETE CASCADE;
ALTER TABLE
courier_db=# \d C_LOGIN;
                        Table "public.c_login"
    Column     |         Type          | Collation | Nullable | Default
---------------+-----------------------+-----------+----------+---------
 loginid       | character varying(20) |           | not null |
 loginusername | character varying(20) |           | not null |
 userpassword  | character varying(20) |           | not null |
 userid        | integer               |           |          |
 roleid        | integer               |           |          |
Indexes:
    "c_login_pkey" PRIMARY KEY, btree (loginid)
    "c_login_loginusername_key" UNIQUE CONSTRAINT, btree (loginusername)
Foreign-key constraints:
    "fk_uid" FOREIGN KEY (userid) REFERENCES c_user(userid)
    "fk_userid" FOREIGN KEY (userid) REFERENCES c_user(userid) ON DELETE CASCADE
    "pk_rid" FOREIGN KEY (roleid) REFERENCES c_role(roleid)
```

5. Delete the roleid from c_login and foreign key constraint

    ALTER TABLE C_LOGIN DROP CONSTRAINT pk_Rid;

    ALTER TABLE C_LOGIN DROP COLUMN Roleid;



```
courier_db=# ALTER TABLE C_LOGIN DROP CONSTRAINT pk_Rid;
ALTER TABLE
courier_db=# \d C_LOGIN;
                        Table "public.c_login"
    Column     |         Type          | Collation | Nullable | Default
---------------+-----------------------+-----------+----------+---------
 loginid       | character varying(20) |           | not null |
 loginusername | character varying(20) |           | not null |
 userpassword  | character varying(20) |           | not null |
 userid        | integer               |           |          |
 roleid        | integer               |           |          |
Indexes:
    "c_login_pkey" PRIMARY KEY, btree (loginid)
    "c_login_loginusername_key" UNIQUE CONSTRAINT, btree (loginusername)
Foreign-key constraints:
    "fk_uid" FOREIGN KEY (userid) REFERENCES c_user(userid)
    "fk_userid" FOREIGN KEY (userid) REFERENCES c_user(userid) ON DELETE CASCADE

courier_db=# ALTER TABLE C_LOGIN DROP COLUMN Roleid;
ALTER TABLE
courier_db=# \d C_LOGIN;
                        Table "public.c_login"
    Column     |         Type          | Collation | Nullable | Default
---------------+-----------------------+-----------+----------+---------
 loginid       | character varying(20) |           | not null |
 loginusername | character varying(20) |           | not null |
 userpassword  | character varying(20) |           | not null |
 userid        | integer               |           |          |
Indexes:
    "c_login_pkey" PRIMARY KEY, btree (loginid)
    "c_login_loginusername_key" UNIQUE CONSTRAINT, btree (loginusername)
Foreign-key constraints:
    "fk_uid" FOREIGN KEY (userid) REFERENCES c_user(userid)
    "fk_userid" FOREIGN KEY (userid) REFERENCES c_user(userid) ON DELETE CASCADE
```

<u>Data Migration</u>

Data migration is the process of transferring the existing data from one system to another or to a new storage system or file format. It is a continuous process that involves streaming real-time data and sharing information across systems. It goes through a series of functions to be loaded onto the target location. It involves the ETL process (extract/transform/load). Data migration is done for several reasons like to enhance performance and competitiveness, upgrade databases, establish new data warehouses or merge the data from an acquisition.

Types of database migration

- An Upgrade to the latest version of DBMS (homogenous migration)
- A switch to a new database from a different provider. Ex: Postgres to MySQL, Oracle to postgres.

Different phases/Steps of migrating data into another database are:-

- ❖ **Planning** – A thorough assessment of the system's operational requirements and how they are adapted to he new environment. We must understand and know whether the data fits within the target system, and check for other problems like incomplete data pieces, inaccuracies, etc before migrating.

- ❖ **Data auditing and profiling** – examining and cleansing the scope of data to be migrated. It aims to detect the possible conflicts, identifying data quality issues and eliminating the duplications and anomalies.

- ❖ **Data backup** – Protecting the data or content before transferring it to prevent the unexpected migration failures or data losses.

- ❖ **Migration design** – It follows the ETL procedure to create scripts for data transition and do the data mapping. It specifies the migration and testing rules and clarifies acceptance criteria and assigns roles and responsibilities across the migration team members.

- ❖ **Execution** – A phase where the data extraction, transformation and loading actually happens. It happens with zero downtime and lowest possible risk of critical failures.

- **Migration Testing** – It is carried across the design, execution and post migration phases. It checks for the data quality and tests the portion of migrated data to fix problems. Frequent testing ensures the safe transit of data elements and high quality and congruence with requirements when entering the target infrastructure.

- **Post migration audit** – This phase ensures that the information has been correctly transported and logged. Before launching the migrated data, the results should be validated with the key clients. The old system retires after the post migration audit.

Some factors to be considered for choosing a database for data migration:

1. Connectivity
2. Scalability
3. Security
4. Speed

Data migration from postgresql to mysql

If we have to migrate to a no-sql database, we would choose My-SQL as it is relatively faster than postgresql and is capable of writing large amounts of data more efficiently and handles concurrency better.

Some limitations of postgres are

- In postgresql there is a **fixed schema** and if we want to add an attribute or data to any one record, we have to add it to all the other records also. But in No-SQL database or My-SQL, the data is stored as document and any number of attributes can be stored and it does not have any fixed schema.
- It is an **open source** – so it does not come with warranty or has no liability or protection and can cause compatibility issues to some users.
- **Slow performance** – It usually performs slower when there is a large number of data stored in the database. Sometimes the query might run

slowly and cause a performance degradation due to its relational database structure.

- Difficulty in upgrading to a newer release.
- Inefficient data replication
- Inefficient architecture for writes
- Issues with table corruption
- The data needs to be exported and replicated to the new version.
- Indexes cannot be used to directly return the results of the query.
- Query execution plans are not cached.
- Postgres has limited high-end options whereas MySQL has a dynamic ecosystem with variants like MariaDB, Percona, Galera.

Team members project contribution:

Name	Contribution	No of hours spent
Ruchita V R	Data migration, Report, writeup, Schema changes.	6
Rajeshwari R	Additional queries, GUI	6
Ramya C	Front end	6