CONCERT RESERVATION SYSTEM

Pranav Kanth
Anbarasan, Hasan
Khwaja, Sreeja Vepa

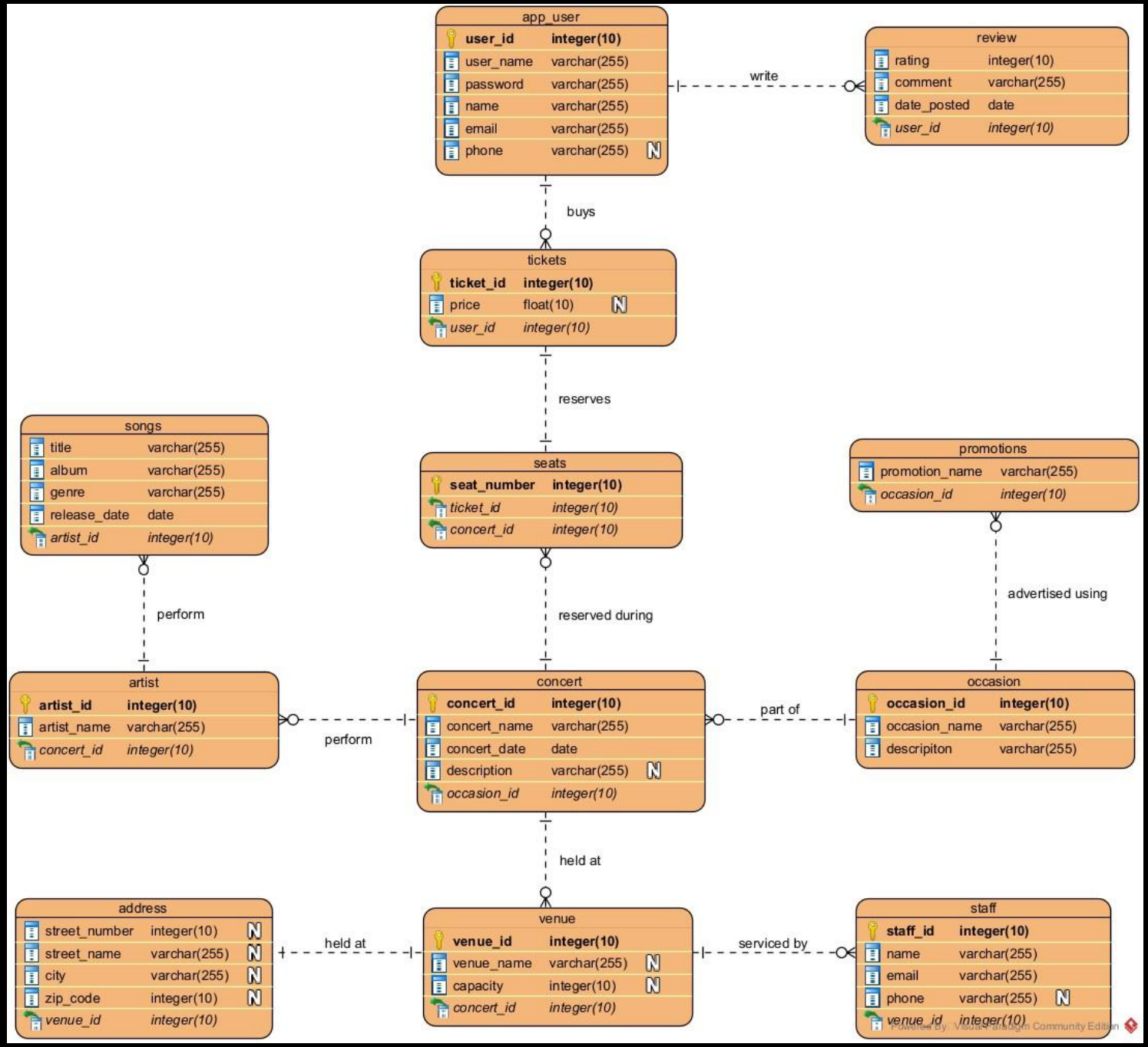
Background

- This system was designed to help users book concert tickets for local artists
- The data was created using the Faker python package
- Used my-sql-connector and pymysql to connect from Python to MySQL

ERD

List of Tables:

App User
Review
Tickets
Seats
Concert
Occasion
Promotions
Artist
Songs
Venue
Address
Staff



Data Collection Methods

- Majority of data was randomly generated using the faker library in python
- Data was generated using predefined list of values

```
def concert_info():
    # swap out connection parameters to match your own system
    cnx_host = 'localhost'
    cnx_user = 'root'
    cnx_pword = ''
    cnx_db = 'music_reservation'

    cnx = pymysql.connect(host = cnx_host, user = cnx_user, password = cnx_pword,
                          db = cnx_db, charset = 'utf8mb4',
                              cursorclass = pymysql.cursors.DictCursor)

    cursor = cnx.cursor()

    # set start and end date parameters
    start_date = datetime.date(2020, 1, 1)
    end_date = datetime.date(2022, 12, 31)

    occasion_id = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

    for concert_id in range(1, 11):
        name = fake.text(max_nb_chars = 15)
        date = fake.date_between_dates(date_start = start_date, date_end = end_date)
        desc = fake.sentence()

        # insert data
        cursor.execute("""
               INSERT INTO concert (concert_id, concert_name, concert_date, description, occassion_id)
               VALUES (%s, %s, %s, %s, %s) """,
               (concert_id, name, date, desc, occasion_id[concert_id - 1]))

     # test query
    query = "select * from concert"
    cursor.execute(query)
    for row in cursor.fetchall():
        print()
        print(row)

    cnx.commit()
    cursor.close()
    cnx.close()

```

Views

- Shows all information of interest to application users
- Shows necessary information needed by staff to help users

```sql
-- the user view shows all information that should be of interest to the average concert viewer
CREATE VIEW user_view AS
    select artist_name, concert_name, concert_date,
        concert.description as concert_description,
        occassion_name, occassion.description as occasion_description,
        venue_name, street_number, street_name, city, zip_code
            from artists
                inner join concert
                    on concert.concert_id = artists.concert_id
                inner join occassion
                    on occassion.occassion_id = concert.occassion_id
                inner join venue
                    on venue.concert_id = concert.concert_id
                inner join address
                    on address.venue_id = venue.venue_id;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

artist_name	concert_name	concert_date	concert_description	occassion_name	occasion_description	venue_name	street_number	street_name	city	zip_code
Erica Guerrero	House walk.	2022-04-27	Position cultural join audience preside...	Million such sit.	Establish voice production three histo...	List rock.	18575	Kim Fall	North Tyler	1968
Blake Lee	House walk.	2022-04-27	Position cultural join audience preside...	Million such sit.	Establish voice production three histo...	List rock.	18575	Kim Fall	North Tyler	1968
Theresa Shaffer	House walk.	2022-04-27	Position cultural join audience preside...	Million such sit.	Establish voice production three histo...	List rock.	18575	Kim Fall	North Tyler	1968
William Jones	Success eight.	2021-01-09	Star every other above less new.	Long two.	Protect collection fire door nearly chal...	Foreign they.	41	Mallory Fields	Ryanborough	64003
Daniel Rios	Success eight.	2021-01-09	Star every other above less new.	Long two.	Protect collection fire door nearly chal...	Foreign they.	41	Mallory Fields	Ryanborough	64003
Ronnie Ryan	Success eight.	2021-01-09	Star every other above less new.	Long two.	Protect collection fire door nearly chal...	Foreign they.	41	Mallory Fields	Ryanborough	64003
Jeremy Fisher	Success eight.	2021-01-09	Star every other above less new.	Long two.	Protect collection fire door nearly chal...	Foreign they.	41	Mallory Fields	Ryanborough	64003
Dawn Campbell MD	Success eight.	2021-01-09	Star every other above less new.	Long two.	Protect collection fire door nearly chal...	Foreign they.	41	Mallory Fields	Ryanborough	64003

```sql
-- purpose of admin view is for staff to be able to see sensitive user information that should be hidden from other users
-- allows staff to see which concert each user is attending as well as what ticket/seat they bought
create view admin_view as
    select app_user.user_id, app_user.password, app_user.name, email, phone,
        tickets.ticket_id, seat.seat_number, tickets.price,
        concert.concert_id,
        concert_name, concert_date,
        occassion.occassion_id, occassion_name,
        venue.venue_id, venue_name
            from concert
                inner join occassion
                    on occassion.occassion_id = concert.occassion_id
                inner join seat
                    on seat.concert_id = concert.concert_id
                inner join tickets
                    on tickets.ticket_id = seat.ticket_id
                inner join venue
                    on venue.concert_id = concert.concert_id
                inner join app_user
                    on app_user.user_id = tickets.user_id;

SELECT * FROM admin_view;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

user_id	password	name	email	phone	ticket_id	seat_number	price	concert_id	concert_name	concert_date	occassion_id	occassion_name	venue_id	venue_name
1	J7bq*J_7HkH#	Dawn Mendoza	johnwhite@example.com	+1-514-353-9	1	1	10	1	House walk.	2022-04-27	1	Million such sit.	1	List rock.
2	jy_*t7Ok+&co	Michael Miller	christina72@example.com	433-797-6719	2	2	15	2	Success eight.	2021-01-09	2	Long two.	2	Foreign they.
3	_aR*3kHY3Nyu	Jonathan Phillips	williamperry@example.org	+1-696-764-6	3	3	20	3	Week put I any.	2021-02-27	3	It training road.	3	Feeling rich.
4	(*4Y8IkdIg08	Mrs. Brittany Vaughan	rhonda46@example.org	518.446.9168	4	4	25	4	Billon system.	2020-07-07	4	Contain total once.	4	Child say.
5	8yGOZKNn%6i&	Jessica Douglas	victoriamoore@example.net	247.919.0445	5	5	30	5	Full structure.	2021-04-20	5	Others fish summer.	5	Learn finish.
6	VT4sKcjo&LV6	Taylor Sweeney	michaelskinner@example.com	(430)975-155	6	6	45	6	Laugh country.	2021-11-08	6	Him data religious.	6	Section first.
7	LsQKqjMr&4UQ	Christopher Hampton	stefanie15@example.net	825.262.8617	7	7	50	7	Old however.	2022-04-04	7	Go region firm firm.	7	Expect rule.
8	)3tWg(LyAqPD	Candace Johnson	justinhenderson@example.com	(688)274-883	8	8	60	8	Child role.	2022-11-29	8	Blue your activity.	8	Owner the.
9	tNx$b@Li^9iX	Maria Wilcox	kwoods@example.org	+1-632-765-7	9	9	70	9	Get sing dark.	2020-12-16	9	Effort glass way.	9	Suggest family.
10	n&8Fz9dB^*#A	Eric Moore	alyssafarrell@example.org	+1-430-609-3	10	10	80	10	Age source.	2021-01-06	10	Should court center.	10	Success bed.

Functions

- Calls the number of songs released in a given year
- Finds the number of songs associated with a given genre
- Finds the number of songs released by a specific artist

```sql
-- function call number of songs released in a year
delimiter $$
create function num_songs_in_year
    (
    target_year int
    )
    returns int
    deterministic reads sql data
    begin
        declare count_song int;

        select count(*) into count_song
            from songs
            where year(release_date) = target_year;

        return count_song;
    end $$
delimiter ;

select num_songs_in_year(2022) as num_songs;

-- function find number of songs with a specific genre
delimiter $$
create function num_songs_with_genre
    (
    target_genre varchar(64)
    )
    returns int
    deterministic reads sql data
    begin
        declare num_genre int;

        select count(*) into num_genre
            from songs
            where songs.genre = target_genre;

        return num_genre;
    end $$
delimiter ;

select num_songs_with_genre('Rock');
```

```sql
-- function find number of songs released by a specific artist
delimiter $$
create function songs_by_artist
    (
    target_artist varchar(64)
    )
    returns int
    deterministic reads sql data
    begin
        declare num_songs int;

        select count(*) into num_songs
            from songs
            inner join artists
                on artists.artist_id = songs.artist_id
            where artists.artist_name = target_artist;

        return num_songs;
    end $$
delimiter ;

select songs_by_artist('Rebecca Cole');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

num_songs
12

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

num_songs_with_genre('Rock')
4

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

songs_by_artist('Rebecca Cole')
4

Stored Procedure 1:
staff_list

Name: staff_list

The name of the
statement. The D

DDL:

```
1  CREATE DEFINER=`root`@`localhost` PROCEDURE `staff_list`(
2  IN venue_name varchar(255))
3  BEGIN
4  -- given venue, find all assigned staff
5  SELECT name
6  FROM staff
7  JOIN venue on staff.venue_id = venue.venue_id
8  WHERE venue.venue_name = venue_name;
9  END
```

Stored Procedure 2:
promotions_for_occasion

Name: promotions_for_occasion

The name of the routine is parsed a
statement. The DDL is parsed auto

DDL:

```
1  CREATE DEFINER=`root`@`localhost` PROCEDURE `promotions_for_occasion`(
2  IN occasion_name varchar(255))
3  BEGIN
4  -- find all promotions for a given occasion
5  SELECT promotion
6  FROM promotions
7  JOIN occassion on occassion.occassion_id = promotions.occassion_id
8  WHERE occassion.occassion_name = occasion_name;
9  END
```

Stored Procedure 3: all_artists_in_concert

Name: all_artists_in_concert

The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:

```
1  CREATE DEFINER=`root`@`localhost` PROCEDURE `all_artists_in_concert`(IN concert_name varchar(255))
2  BEGIN
3  -- given concert name find all artists assigned to concert
4  SELECT artists.artist_name
5  FROM artists
6  JOIN concert on artists.concert_id = concert.concert_id
7  WHERE concert.concert_name = concert_name;
8  END
```

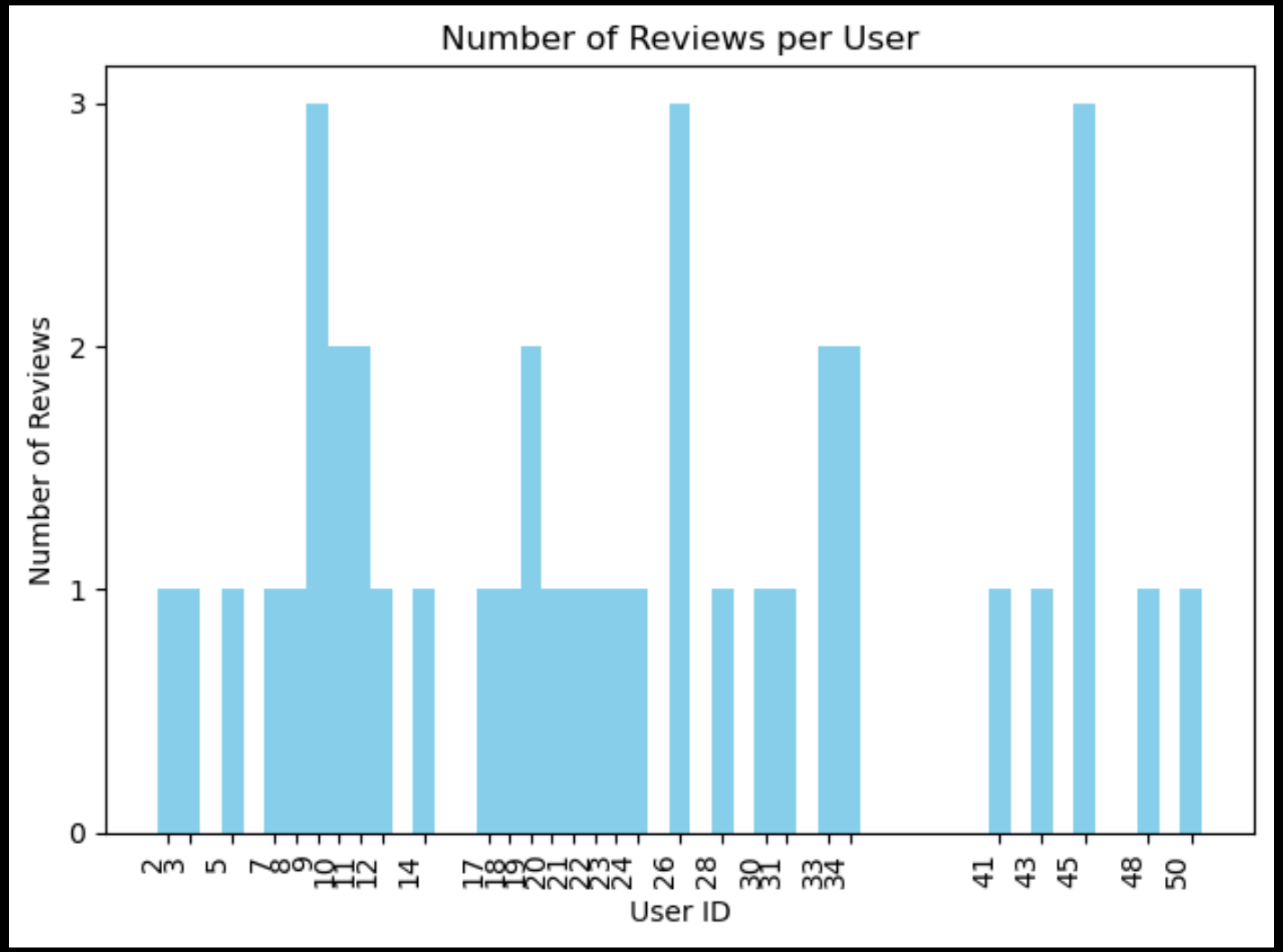
Trigger

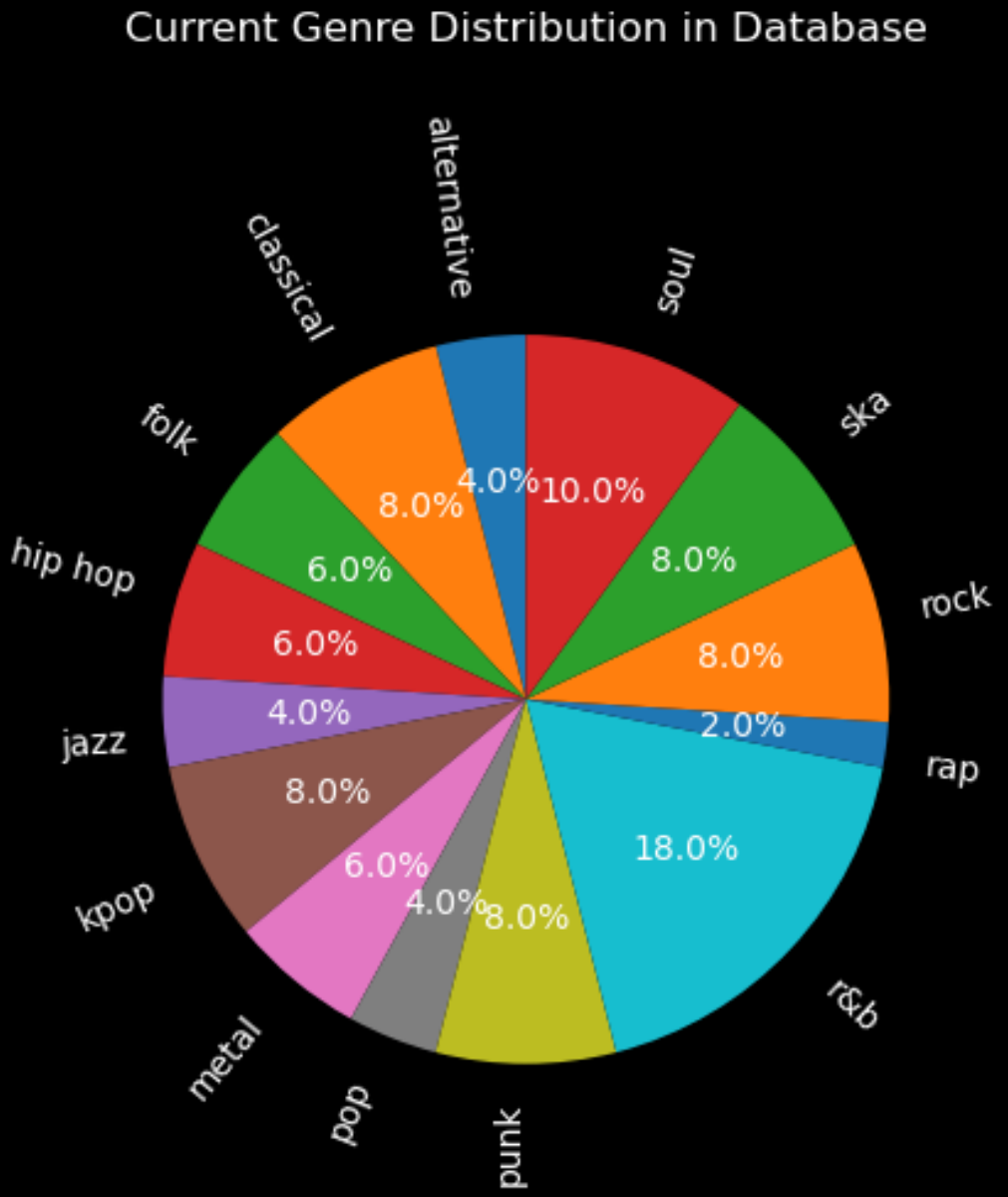
- Before Insert Update
- Checks if capacity of concert is filled before letting user buy a ticket
- Gives error message: 'No more seats are left' if capacity is full

```
/*!50003 CREATE*/ /*!50017 DEFINER=`root`@`localhost`*/ /*!50003 TR
DECLARE num_tickets INT;
DECLARE capacity_var INT;
DECLARE msg varchar(128);

SELECT COUNT(*) INTO num_tickets
FROM seat
INNER JOIN concert on concert.concert_id = seat.concert_id
AND concert.concert_id = new.concert_id;

SELECT capacity INTO capacity_var
FROM venue
INNER JOIN concert on concert.concert_id = venue.concert_id
WHERE concert.concert_name = concert_name;

IF capacity_var = num_of_tickets
THEN SET msg = 'No more seats are left';
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = msg;
END IF;
END */;;
DELIMITER ;
```

Figures/Reports

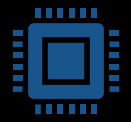
Out of 50 users, 29 posted reviews





Front end

The graphical user interface (GUI) of the application has been constructed utilizing the Tkinter library, a python framework dedicated to the GUI development.



Tkinter offers a high degree of extensibility, allowing users to create custom widgets and modify existing ones to suit the specific design requirements of their applications. This flexibility empowers us to make visually appealing interfaces.

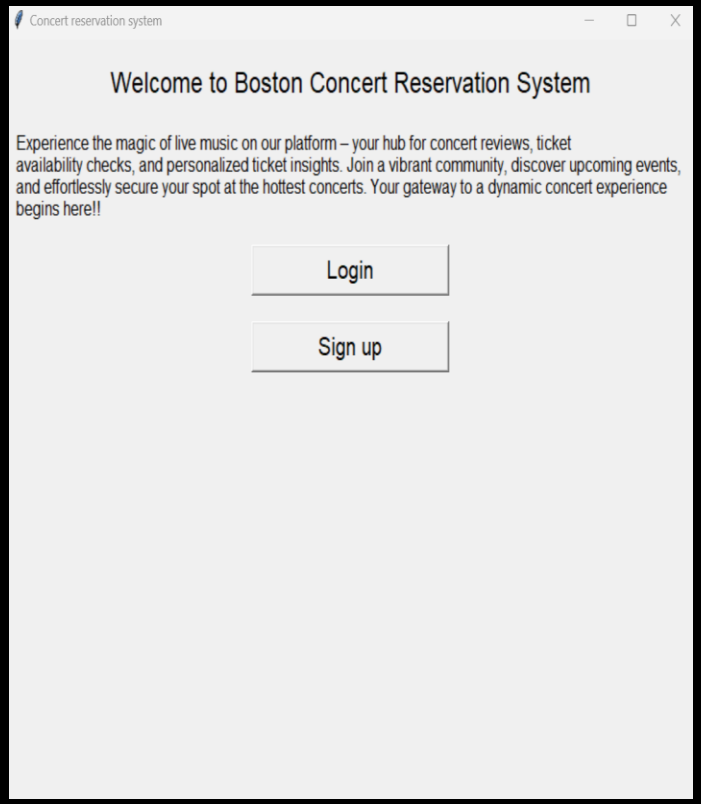
Tkinter easily integrates with the MySQL database through the utilization of the ***mysql-connector*** *package. This powerful combination facilitates efficient communication between the front-end interface built with Tkinter and the MySQL database, enabling seamless data retrieval, storage, and manipulation.*

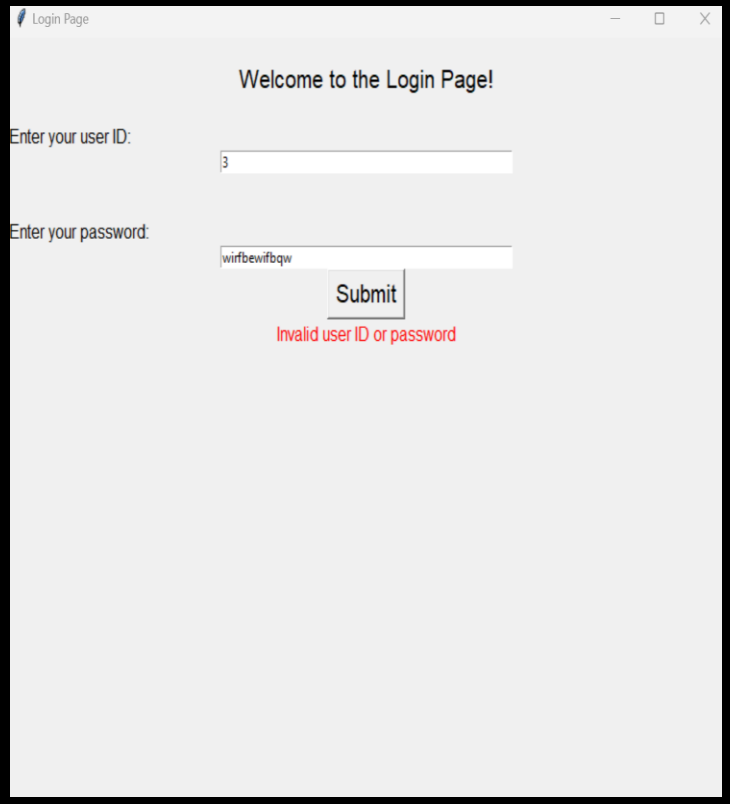
Libraries Used:

- Tkinter
- **MySqlConnector** is a Python module that serves as a bridge between Python applications, such as those employing Tkinter for front-end development, and MySQL databases.
- The **datetime** module has been employed to automatically generate default dates during the ticket booking process initiated by the user.
- The **random** module has been utilized to generate randomized values for specific attributes in the database when users complete designated pages in the frontend.

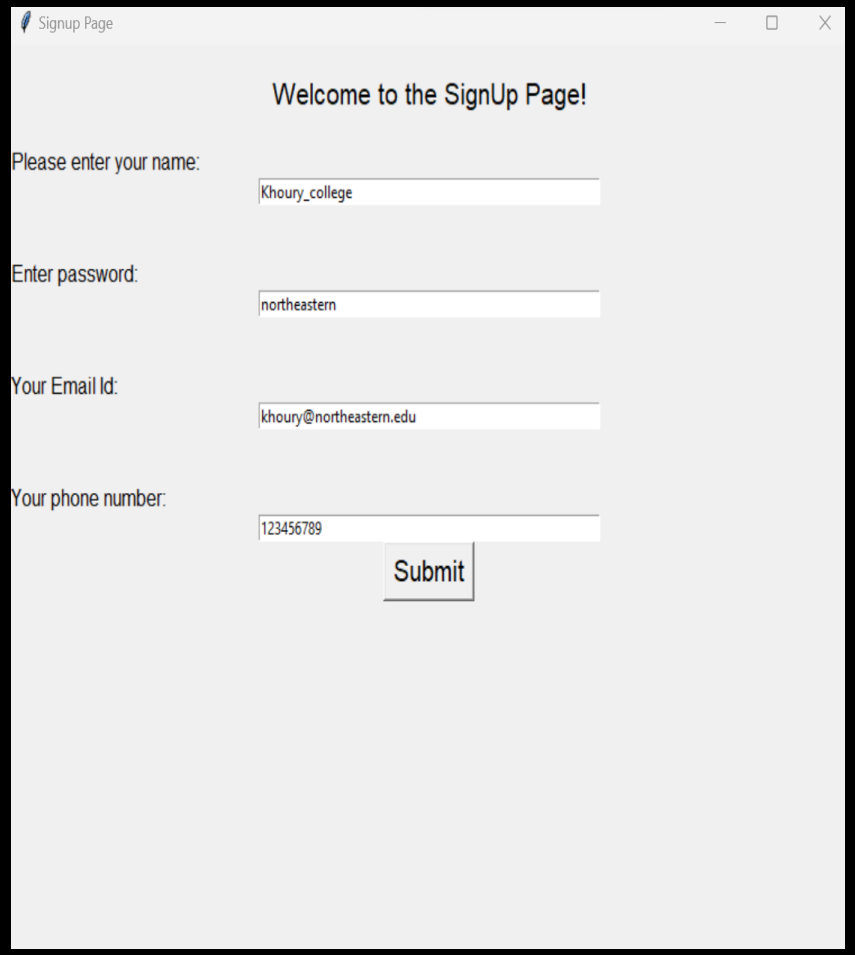
Login/ Signup Page

The first page features options for user login and signup. Upon selecting the login option, users are prompted to enter their user ID and password. New users are required to complete the signup process, and upon successful data storage in the database, they gain access to the login page. Upon entering correct credentials on the login page, users are redirected to the main page.





Continued



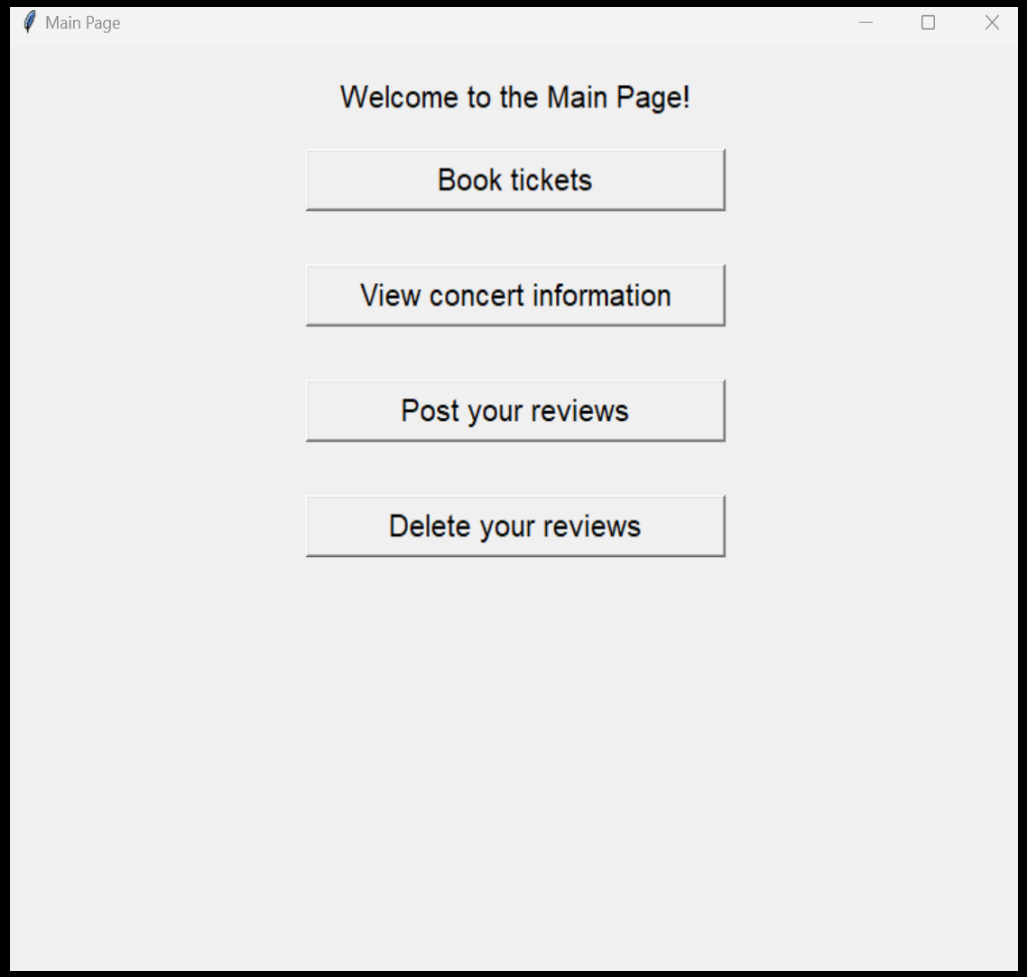
52	epiwfn24pifn	ewofneofn	e2jofon2f 2p	23242598989
53	pass	northeastern	pass@gmail.com	2234522344
54	qwerty	boston	boston@northeastern.edu	09737455355
55	zxcv	rogerog	34pn43pigin	534636
56	northeastern	khoury_college	khoury@northeastern.edu	123456789
NULL	NULL	NULL	NULL	NULL

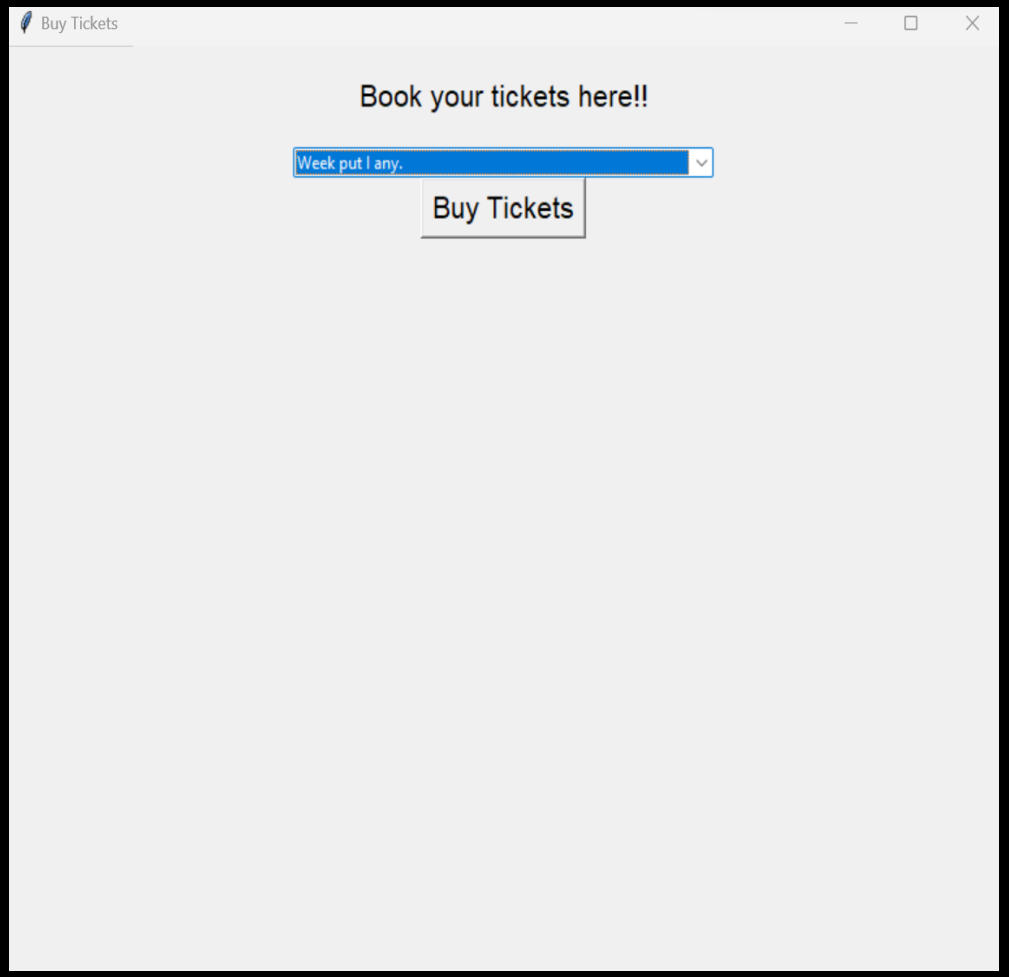
Main page

- Upon successful authentication, users are directed to the main page, where they can purchase tickets and access comprehensive concert details, including the concert name, venue, and address.

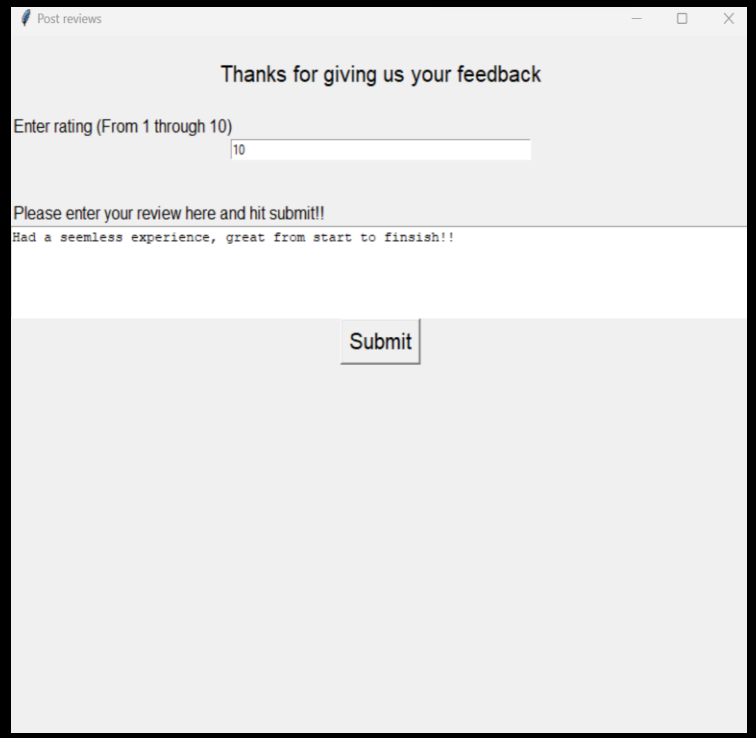
- The interface further enables users to share their reviews about their experiences.

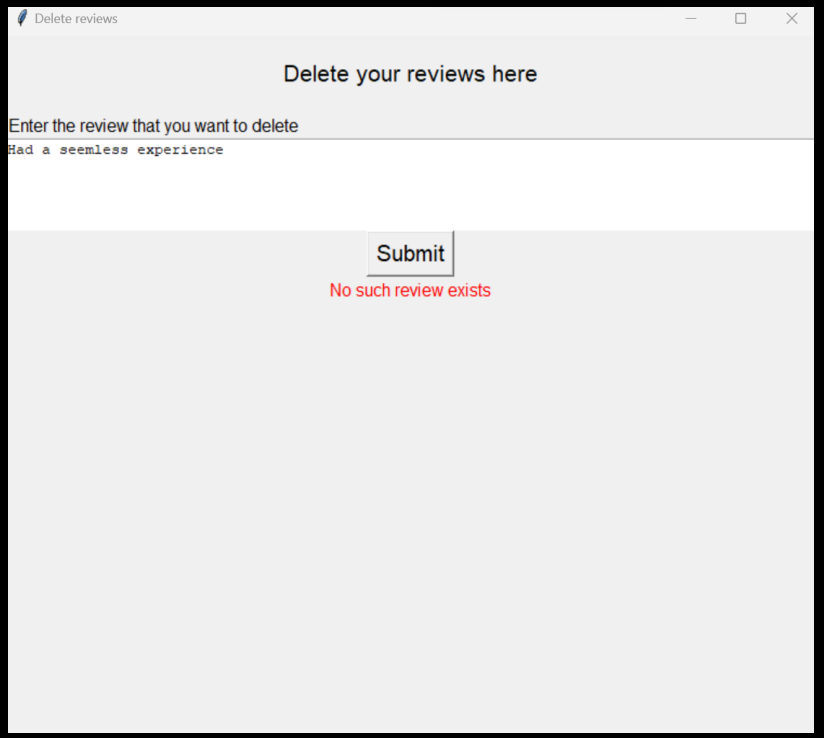
- In the event of a user opting to delete a review, they can click the "Delete Your Reviews" button, enter the relevant review details, and, if the review exists, it will be removed from the database; otherwise, an error message stating "No such review exists" will be displayed.


Main Page
Welcome to the Main Page!
Book tickets
View concert information
Post your reviews
Delete your reviews


Buy Tickets
Book your tickets here!!
Week put I any.
Buy Tickets

	ticket_id	price	user_id
	11	74....	53
	12	65....	52
	13	54....	49
	14	60....	54
	15	77....	56
	NULL	NULL	NULL





	10	Bank you social budget thing scene.	2022-09-02	28
	10	Sound page time modern himself both million.	2022-07-10	2
	10	Had a seemless experience, great from start...	2023-12-07	56
	9	Add alone policy gun certain.	2022-05-04	45
	9	Enter serve baby really.	2020-12-21	18
	9	great concert!!	2023-12-04	1

	rating	comment	date_posted	user_id
▶	10	Size wide second cause.	2022-04-02	10
	10	Truth protect learn matter more.	2022-06-21	20
	10	Artist cover song message source.	2022-04-29	9
	10	Bank you social budget thing scene.	2022-09-02	28
	10	Sound page time modern himself both million.	2022-07-10	2
	9	Add alone policy gun certain.	2022-05-04	45

Summary

Leveraging the high-level programming language Python, specifically utilizing Tkinter, we have engineered an application that showcases seamless interaction with the MySQL database.

This application allows the user to execute queries on tables and facilitates the insertion, updating, and deletion of data within designated tables.

We have 3 stored procedures, 3 functions, 1 trigger, and 2 views in this project.

A significant portion of our time was spent on data creation.