E-ticket Reservation System

CPS 510

11/26/23

Min Cho 500945255

Table of Contents

Assignment 1 : Proposal

Topic & Purpose:
Our chosen topic is an online E-ticket Reservation DBMS. This service will be created for users to access and book tickets for available events.
It will also serve as a portal for event hosts to upload and manage their event information (location, time, etc.). Administrators will have root access to all information in the database and can manage details accordingly.

System Functions:
System functions are divided into three different perspectives as there are users, hosts, and administrators using this program.
The program is able to search and create the event using the criteria below:

- Type of events : Movie / Concert / Sports / Art
- Name of the events
- Age
- Price
- Time
- Location
- Capacity

It then recommends the users few events that fit their interests or tells hosts and administrators how many and who reserved the events.

Some Queries and Update Operations to apply to the DB:

a) (query) List names of all persons who bought a ticket for a specific event and its details.
b) (query) Retrieve all information of a specific person who purchased a ticket
c) (update) Insert a new customer who purchased a ticket into the Database
d) (update) Remove a customer who decided to cancel their ticket from the Database
e) (update) Add more events

All relationships among the records of the DB:

a) Each EVENT has their own GENRE/SPORTS, PRICE, LOCATION, TIME, AGE LIMIT, and CAPACITY
b) All the events can be categorized by criteria above.

Examples of Integrity Constraints:

a) All customers should have a unique identifier
b) All events should have a unique identifier
c) All customers' age at an event must be higher than the age limit
d) The number of customers for a specific event must be lower than the Capacity for that event
e) The program will only recommend events that have common time, and age limit with their search record.

Different Users, what would they need from the Database:

a) Administrator:
    - Has root access to database
    - Ability to manage and edit information on the database
    - Can access and check who bought what ticket
b) Event Host:
    - Can create and upload new events to the database
    - Ability to manage events of self
    - Can access and check only how many people bought the ticket
c) Customer:
    - Access to frontend side of database
    - Ability to book tickets
    - Ability to cancel tickets
    - Search through database for specific event types
    - System recommends similar events after they search

Example Tables for DB:

Events

Name	Code	Genre/Sports	Time	Price	Location	Capacity	Age limit
Avengers	MV	AC	1800	20.00	T	1000	15

Types

TYPE	Code
Movie	MV
Concert	CC
Sports	SP
Art	AT

Genre

TYPE	Code
Action	AC
Adventure	AT
Comedy	CD
Romance	RO
Horror	HR
Mystery	MY
Science Fiction	SF
Sports	**Code**
Football	FB
Baseball	BB
Basketball	BK
Hockey	HO

Customer

Name	Age	Purchased	User Name
Bob	15	Avengers	

CODE	EVENT
MV	1
MV	1
MV	1
MV	1

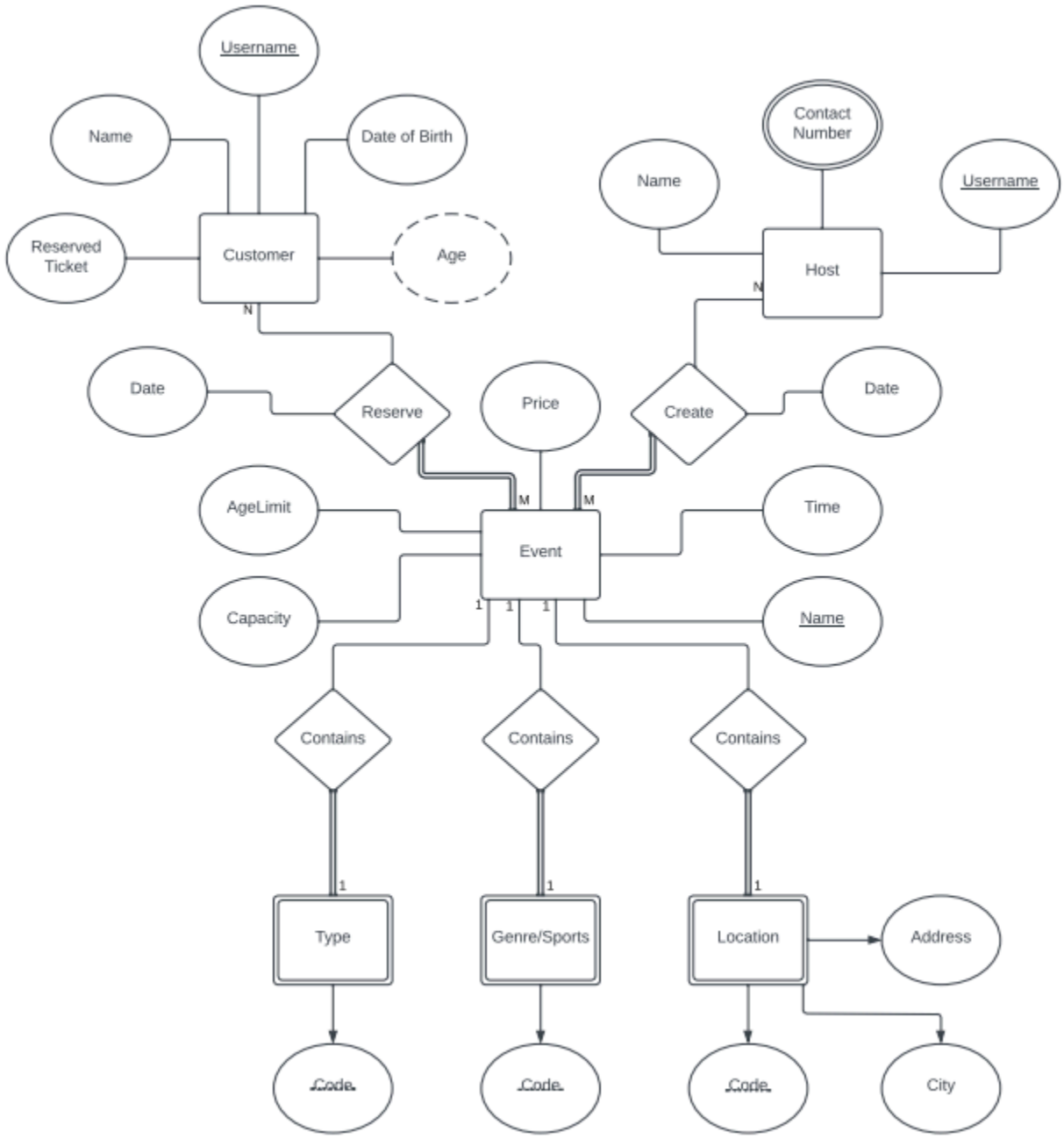
LOCATION

ADDRESS	City	City code
123 Street Avenue	Toronto	T
123 Waterloo Av	Hamilton	H
543 Victoria Street	North York	N
3929 Happy street	Mississauga	M
1124 Podium Street	Scarbrough	S

Conclusion

Overall, the E-ticket Reservation System will have one EVENTS data file that contains and will call other data files such as TYPES, GENRES/SPORTS, etc. Thus, EVENTS are able to be searched and sorted with the criteria listed above. It then allows the program to recommend users events that are related to their interests, allows the hosts to check how many people have reserved their events, and allows administrators to check who reserved what events and how many people have reserved using customers' name or even the title of the events.

Assignment 2: ER Diagram



Assignment 3 : Entities

Strong Entities

TMU 12C | CUSTOMER

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	CUSTOMERUSERNAME	VARCHAR2(30 BYTE)	No	(null)	1	(null)
2	CUSTOMERNAME	VARCHAR2(30 BYTE)	Yes	(null)	2	(null)
3	CUSTOMERDATEOFBIRTH	TIMESTAMP(6)	Yes	(null)	3	(null)
4	CUSTOMERRESERVED	VARCHAR2(30 BYTE)	Yes	(null)	4	(null)

TMU 12C | CUSTOMER

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQ

Sort.. | Filter:

	CUSTOMERUSERNAME	CUSTOMERNAME	CUSTOMERDATEOFBIRTH	CUSTOMERRESERVED
1	john_doe	John Doe	22-03-03 00:00:00.000000000	Ticket01

TMU 12C | EVENT

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | S

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	EVENTNAME	VARCHAR2(30 BYTE)	No	(null)	1	(null)
2	EVENTTIME	TIMESTAMP(6)	Yes	(null)	2	(null)
3	EVENTPRICE	NUMBER(10,2)	Yes	(null)	3	(null)
4	EVENTCAPACITY	NUMBER(38,0)	Yes	(null)	4	(null)
5	EVENTAGELIMIT	NUMBER(38,0)	Yes	(null)	5	(null)

TMU 12C | EVENTHOST

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	EVENTHOSTUSERNAME	VARCHAR2(30 BYTE)	No	(null)	1	(null)
2	EVENTHOSTNAME	VARCHAR2(30 BYTE)	Yes	(null)	2	(null)
3	EVENTHOSTEVENTSLIST	VARCHAR2(30 BYTE)	Yes	(null)	3	(null)
4	EVENTHOSTCONTACT	VARCHAR2(200 BYTE)	Yes	(null)	4	(null)

TMU 12C | EVENTHOST

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes

Sort.. | Filter:

	EVENTHOSTUSERNAME	EVENTHOSTNAME	EVENTHOSTEVENTSLIST	EVENTHOSTCONTACT
1	event_host1	Event Host 1	Kanye Concert	647-123-4567

TMU 12C | EVENT

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Sort.. | Filter:

	EVENTNAME	EVENTTIME	EVENTPRICE	EVENTCAPACITY	EVENTAGELIMIT
1	Kanye Concert	23-09-30 10:00:00.000000000	50	1000	19

Weak Entities

TMU 12C | EVENTLOCATION

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	EVENTLOCATIONADDRESS	VARCHAR2(30 BYTE)	No	(null)	1	(null)
2	EVENTLOCATIONCITY	VARCHAR2(30 BYTE)	No	(null)	2	(null)
3	EVENTLOCATIONCODE	VARCHAR2(5 BYTE)	No	(null)	3	(null)

TMU 12C | EVENTLOCATION

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | D

Sort.. | Filter:

	EVENTLOCATIONADDRESS	EVENTLOCATIONCITY	EVENTLOCATIONCODE
1	123 Dundas St	Toronto	T

TMU 12C | EVENTGENRE

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | S

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	EVENTGENRENAME	VARCHAR2(30 BYTE)	No	(null)	1	(null)
2	EVENTGENRECODE	VARCHAR2(2 BYTE)	No	(null)	2	(null)

TMU 12C | EVENTGENRE

Columns | Data | Model | Constraints | Grants | Statistics | Trig

Sort.. | Filter:

	EVENTGENRENAME	EVENTGENRECODE
1	Basketball	BK

TMU 12C | EVENTTYPE

Columns | Data | Model | Constraints | Grants | Statistics | T

Sort.. | Filter:

	EVENTTYPENAME	EVENTTYPECODE
1	Concert	CC

TMU 12C | EVENTTYPE

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | S

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	EVENTTYPENAME	VARCHAR2(30 BYTE)	No	(null)	1	(null)
2	EVENTTYPECODE	VARCHAR2(2 BYTE)	No	(null)	2	(null)

Relationships

TMU 12C | RESERVE

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	RESERVEDATE	TIMESTAMP(6)	Yes	(null)	1	(null)
2	CUSTOMERUSERNAME	VARCHAR2(30 BYTE)	No	(null)	2	(null)
3	EVENTNAME	VARCHAR2(30 BYTE)	No	(null)	3	(null)

TMU 12C | CREATEEVENT

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	CREATEEVENTDATE	TIMESTAMP(6)	Yes	(null)	1	(null)
2	EVENTHOSTUSERNAME	VARCHAR2(30 BYTE)	No	(null)	2	(null)
3	EVENTNAME	VARCHAR2(30 BYTE)	No	(null)	3	(null)

```
/*Strong Entity*/
CREATE TABLE Customer (
   CustomerUserName varchar (30) NOT NULL,
   CustomerName VARCHAR(30),
         CustomerDateOfBirth TIMESTAMP,
   /* CustomerAge INT, */
   CustomerReserved varchar (30),
         PRIMARY KEY (CustomerUserName));

CREATE TABLE EventHost (
   EventHostUsername VARCHAR(30) NOT NULL,
   EventHostName VARCHAR(30),
   EventHostEventsList VARCHAR(30),
   EventHostContact VARCHAR(200),
   PRIMARY KEY (EventHostUsername));

CREATE TABLE Event(
   EventName VARCHAR(30) NOT NULL,
   EventTime TIMESTAMP,
   EventPrice DECIMAL(10,2),
   EventCapacity INT,
   EventAgeLimit INT,
   PRIMARY KEY (EventName));

/*Relationship*/
Create Table Reserve(
   ReserveDate TIMESTAMP,
   CustomerUsername varchar(30) REFERENCES Customer (CustomerUsername),
   EventName varchar(30)REFERENCES Event (EventName),
   primary key (CustomerUsername,EventName) );

CREATE TABLE CreateEvent (
         CreateEventDate TIMESTAMP,
         EventHostUsername VARCHAR(30) NOT NULL,
         EventName VARCHAR(30) NOT NULL,
         FOREIGN KEY (EventHostUsername) REFERENCES EventHost(EventHostUsername),
         FOREIGN KEY (EventName) REFERENCES Event(EventName));


/*Weak Entity */
CREATE TABLE EventType (
   EventTypeName varchar(30) NOT NULL,
   EventTypeCode varchar(2) NOT NULL,
   Unique (EventTypeCode));
```

```
CREATE TABLE EventGenre (
   EventGenreName varchar(30) NOT NULL,
   EventGenrecode varchar(2) NOT NULL,
   Unique (EventGenrecode));

CREATE TABLE EventLocation (
   EventLocationAddress VARCHAR(30) NOT NULL,
   EventLocationCity VARCHAR(30) NOT NULL,
   EventLocationCode VARCHAR(5) NOT NULL,
   Unique (EventLocationCode));

INSERT INTO Event (EventName, EventTime, EventPrice, EventCapacity, EventAgeLimit)
VALUES ('Kanye Concert', '2023-09-30 10:00:00', 50.00, 1000, 19);

INSERT INTO Customer (CustomerUserName, CustomerName, CustomerDateOfBirth,
CustomerReserved)
VALUES ('john_doe', 'John Doe', '22-MAR-03', 'Ticket01');

INSERT INTO EventHost (EventHostUsername, EventHostName, EventHostEventsList,
EventHostContact)
VALUES ('event_host1', 'Event Host 1', 'Kanye Concert', '647-123-4567');

INSERT INTO EventType (EventTypeName, EventTypeCode)
VALUES ('Concert', 'CC');

INSERT INTO EventGenre (EventGenreName, EventGenreCode)
VALUES ('Basketball', 'BK');

INSERT INTO EventLocation (EventLocationAddress, EventLocationCity, EventLocationCode)
VALUES ('123 Dundas St', 'Toronto', 'T');
```

Assignment 4: Queries

/*1 List all attributes of events in Concert type */

```
SELECT *
FROM Event
WHERE EventName IN (
        SELECT EventName
        FROM containstype
        WHERE EVENTTYPECODE IN (
    SELECT EVENTTYPECODE
    From Eventtype
    where EVENTTYPENAME = 'Concert') );
```

EVENTNAME	EVENTTIME	EVENTPRICE	EVENTCAPACITY	EVENTAGELIMIT
Taylor Swift Concert	23-05-20 10:00:00.000000000	250	2000	16
Kanye Concert	23-09-30 10:00:00.000000000	50	1000	19

/*2 List all attributes of events in Romance genre */

```
SELECT *
FROM Event
WHERE EventName IN (
        SELECT EventName
        FROM containsgenre
        WHERE EVENTGENRECODE IN (
    SELECT EVENTGENRECODE
    From EVENTGENRE
    where EVENTGENRENAME = 'Romance') );
```

EVENTNAME	EVENTTIME	EVENTPRICE	EVENTCAPACITY	EVENTAGELIMIT
Kanye Concert	23-09-30 10:00:00.000000000	50	1000	19

/*3 List all attributes of events happening in Toronto */

```
SELECT *
FROM Event
WHERE EventName IN (
        SELECT EventName
        FROM CONTAINSLOCATION
        WHERE EventLocationAddress IN (
    SELECT EventLocationAddress
    From EVENTLOCATION
    where EVENTLOCATIONCITY = 'Toronto') );
```

EVENTNAME	EVENTTIME	EVENTPRICE	EVENTCAPACITY	EVENTAGELIMIT
Taylor Swift Concert	23-05-20 10:00:00.000000000	250	2000	16
Kanye Concert	23-09-30 10:00:00.000000000	50	1000	19

/* 4 List all attributes of events which John Doe reserved */

```
SELECT *
FROM Event
WHERE EventName IN (
        SELECT EventName
        FROM Reserve
        WHERE CustomerUsername = 'john_doe'
);
```

EVENTNAME	EVENTTIME	EVENTPRICE	EVENTCAPACITY	EVENTAGELIMIT
Kanye Concert	23-09-30 10:00:00.000000000	50	1000	19
Taylor Swift Concert	23-05-20 10:00:00.000000000	250	2000	16

/* 5 List all attributes of events with event host “event_host1” */

```
SELECT *
FROM Event
WHERE EventName IN (
        SELECT EventName
        FROM CreateEvent
        WHERE EventHostUsername = 'event_host1'
);
```

EVENTNAME	EVENTTIME	EVENTPRICE	EVENTCAPACITY	EVENTAGELIMIT
Kanye Concert	23-09-30 10:00:00.000000000	50	1000	19
Taylor Swift Concert	23-05-20 10:00:00.000000000	250	2000	16

/* 6 List concert event with largest EVENTCAPACITY */

```
SELECT *
FROM Event
WHERE EventName IN (
        SELECT EventName
        FROM (
        SELECT EventName
        FROM Event
        ORDER BY EventCapacity DESC
        )
        WHERE ROWNUM = 1
);
```

EVENTNAME	EVENTTIME	EVENTPRICE	EVENTCAPACITY	EVENTAGELIMIT
Taylor Swift Concert	23-05-20 10:00:00.000000000	250	2000	16

/*7 List all attributes of events with the age limit of 19 */

```
SELECT *
FROM Event
WHERE EventAgeLimit = 19;
```

	EVENTNAME	EVENTTIME	EVENTPRICE	EVENTCAPACITY	EVENTAGELIMIT
1	Kanye Concert	23-09-30 10:00:00.000000000	50	1000	19

/*8 List all concert events according to ticket prices (ascending) */

```
SELECT Event.*
FROM Event
JOIN ContainsType ON Event.EventName = ContainsType.EventName
WHERE ContainsType.EventTypeCode = 'CC'
ORDER BY Event.EventPrice ASC;
```

	EVENTNAME	EVENTTIME	EVENTPRICE	EVENTCAPACITY	EVENTAGELIMIT
1	Kanye Concert	23-09-30 10:00:00.000000000	50	1000	19
2	Taylor Swift Concert	23-05-20 10:00:00.000000000	250	2000	16

/*9 List how many customers have purchased tickets for each Concert */

```
SELECT EventName, COUNT(CustomerUsername) AS NumberReserved
FROM Reserve
GROUP BY EventName;
```

	EVENTNAME	NUMBERRESERVED
1	Taylor Swift Concert	1
2	Kanye Concert	1

Advanced Queries

/* 1. List the event type that contains the events with age limit < 19 */

```
select EVENTTYPECODE as limit_19_code
from CONTAINSTYPE
where EVENTNAME in (
   select EVENTNAME
   from EVENT
   where EVENTAGELIMIT < 19);
```

	LIMIT_19_CODE
1	CC
2	SP
3	SP

/* 1.5 Remove any redundant results from the previous example */

```
select DISTINCT EVENTTYPECODE as limit_19_code
from CONTAINSTYPE
where EVENTNAME in (
   select EVENTNAME
   from EVENT
   where EVENTAGELIMIT < 19);
```

	LIMIT_19_CODE
1	SP
2	CC

/* 2 List name of the event that has capacity bigger than 1000

```
select eventname
from event e
where  eventcapacity >1000;
```

```
EVENTNAME
------------------------------
Taylor Swift Concert
Blue Jays
Raptors
```

/*2.5 But are not in type of concert

```
select eventname
from event e
where  eventcapacity >1000
   MINUS
      (select c.eventname
       from CONTAINSTYPE c
       where c.eventtypecode = 'CC');
```

```
EVENTNAME
------------------------------
Blue Jays
Raptors
```

/* 3 List name of event with the corresponding event type name */

```
SELECT Event.EventName, EventType.EventTypeName
FROM Event
JOIN ContainsType ON Event.EventName = ContainsType.EventName
JOIN EventType ON ContainsType.EventTypeCode = EventType.EventTypeCode;
```

	EVENTNAME	EVENTTYPENAME
1	Taylor Swift Concert	Concert
2	Kanye Concert	Concert
3	Raptors	Sports
4	Blue Jays	Sports

Views

/*1 Creates a VIEW for events of type Concert ‘CC’ */

```
CREATE VIEW ConcertEvents AS
SELECT *
FROM Event
WHERE EventName IN (
        SELECT EventName
        FROM ContainsType
        WHERE EventTypeCode = 'CC'
);
SELECT *
FROM ConcertEvents;
```

EVENTNAME	EVENTTIME	EVENTPRICE	EVENTCAPACITY	EVENTAGELIMIT
Kanye Concert	23-09-30 10:00:00.000000000	50	1000	19
Taylor Swift Concert	23-05-20 10:00:00.000000000	250	2000	16

/*2 Creates a VIEW for events happening at 123 Dundas St */

```
CREATE VIEW EventsByLocation AS
SELECT EventLocationAddress, Event.EventName AS EventName
FROM ContainsLocation
JOIN Event ON ContainsLocation.EventName = Event.EventName
WHERE EventLocationAddress = '123 Dundas St';
SELECT *
FROM EventsByLocation;
```

EVENTLOCATIONADDRESS	EVENTNAME
123 Dundas St	Kanye Concert

/*3 Creates a VIEW for all the events with its host and customer*/

```
CREATE VIEW whowho  AS
SELECT c.eventname, c.eventhostusername, r.customerusername
FROM CREATEEVENT c , Reserve r
WHERE c.eventname = r.eventname;
select *
from whowho ;
```

EVENTNAME	EVENTHOSTUSERNAME	CUSTOMERUSERNAME
Blue Jays	event_host1	john_doe
Kanye Concert	event_host1	john_doe
Raptors	event_host1	john_doe
Taylor Swift Concert	event_host1	john_doe

/* 4 Creates a VIEW for all events showing their capacity and current number reservations */

```
CREATE VIEW EventReservations AS
SELECT Event.EventName, Event.EventCapacity, COUNT(Reserve.CustomerUsername) AS
ReservationCount
FROM Event
JOIN Reserve ON Event.EventName = Reserve.EventName
GROUP BY Event.EventName, Event.EventCapacity;

SELECT * FROM EventReservations;
```

	EVENTNAME	EVENTCAPACITY	RESERVATIONCOUNT
1	Kanye Concert	1000	1
2	Raptors	2000	1
3	Taylor Swift Concert	2000	1
4	Blue Jays	2000	1

Assignment 5 : Exists / Union / Minus / Count / Group By

Exists

```
/* List the names of all events that have reservations */
SELECT EventName
FROM Event e
WHERE EXISTS (
   SELECT 1
   FROM Reserve r
   WHERE r.EventName = e.EventName
);
```

	EVENTNAME
1	Blue Jays
2	Kanye Concert
3	Raptors
4	Taylor Swift Concert

Union

```
/* List all genres and types and states what they are */
SELECT EventType.*, 'Type' AS TypeORGenre
FROM EventType

UNION

SELECT EventGenre.*, 'Genre' AS TypeORGenre
FROM EventGenre;
```

	EVENTTYPENAME	EVENTTYPECODE	TYPEORGENRE
1	Action	AC	Genre
2	Adventure	AT	Genre
3	Art	AT	Type
4	Baseball	BB	Genre
5	Basketball	BK	Genre
6	Comedy	CD	Genre
7	Concert	CC	Type
8	Football	FB	Genre
9	Hockey	HO	Genre
10	Horror	HR	Genre
11	Movie	MV	Type
12	Mystery	MY	Genre
13	Romance	RO	Genre
14	Science-Fiction	SF	Genre
15	Sports	SP	Type

Minus

```
/* List name of the event that has capacity bigger than 1000 But are not in type of concert */
select eventname
from event e
where  eventcapacity >1000
   MINUS
      (select c.eventname
       from CONTAINSTYPE c
       where c.eventtypecode = 'CC');
```

```
EVENTNAME
------------------------------
Taylor Swift Concert
Blue Jays
Raptors


EVENTNAME
------------------------------
Blue Jays
Raptors
```

Count

```
/* Identifies event with highest number of reservations */
SELECT EventName, ReservationCount
FROM (
	SELECT EventName, COUNT(*) AS ReservationCount
	FROM Reserve
	GROUP BY EventName
	ORDER BY ReservationCount DESC
)
WHERE ROWNUM = 1;
```

Group By

```
/* Identifies event location with highest event count and finds average price for event at that
location */
SELECT cl.EventLocationAddress, COUNT(cl.EventName) AS EventCount, AVG(e.EventPrice) AS
AverageEventPrice
FROM ContainsLocation cl
JOIN Event e ON cl.EventName = e.EventName
GROUP BY cl.EventLocationAddress
```

CMD Shell

Menu.sh

```
#!/bin/sh
MainMenu()
{
 while [ "$CHOICE" != "START" ]
 do
     echo "=================================================================="
     echo "| Oracle All Inclusive Tool|"
     echo "| Main Menu - Select Desired Operation(s):|"
     echo "| <CTRL-Z Anytime to Enter Interactive CMD Prompt>|"
     echo "------------------------------------------------------------------"
     echo " $IS_SELECTEDM M) View Manual"
     echo " "
     echo " $IS_SELECTED1 1) Drop Tables"
     echo " $IS_SELECTED2 2) Create Tables"
     echo " $IS_SELECTED3 3) Populate Tables"
     echo " $IS_SELECTED4 4) Query Tables"
     echo " "
     echo " $IS_SELECTEDE E) End/Exit"
     echo "Choose: "
     read CHOICE
     if [ "$CHOICE" == "0" ]
     then
         echo "Nothing Here"
     elif [ "$CHOICE" == "1" ]
     then
         bash drop_tables.sh
         Pause
     elif [ "$CHOICE" == "2" ]
     then
         bash create_tables.sh
         Pause
     elif [ "$CHOICE" == "3" ]
     then
         bash populate_tables.sh
         Pause
     elif [ "$CHOICE" == "4" ]
     then
         bash queries.sh
         Pause
     elif [ "$CHOICE" == "E" ]
     then
         exit
     fi
 done
}
#--COMMENTS BLOCK--
# Main Program
#--COMMENTS BLOCK--
ProgramStart()
{
     while [ 1 ]
     do
       MainMenu
     done
```

```
=================================================================
| Oracle All Inclusive Tool|
| Main Menu - Select Desired Operation(s):|
| <CTRL-Z Anytime to Enter Interactive CMD Prompt>|
-----------------------------------------------------------------
  M) View Manual

  1) Drop Tables
  2) Create Tables
  3) Populate Tables
  4) Query Tables

  E) End/Exit
Choose:
```

drop_tables.sh

```
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64
"m32cho/@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF

drop table containstype CASCADE CONSTRAINTS;
drop table containsgenre CASCADE CONSTRAINTS;
drop table containslocation CASCADE CONSTRAINTS;
drop table createevent CASCADE CONSTRAINTS;
drop table eventgenre CASCADE CONSTRAINTS;
drop table eventhost CASCADE CONSTRAINTS;
drop table eventlocation CASCADE CONSTRAINTS;
drop table eventtype CASCADE CONSTRAINTS;
drop table reserve CASCADE CONSTRAINTS;
drop table customer CASCADE CONSTRAINTS;
drop table event CASCADE CONSTRAINTS;
exit;
EOF
```

```
SQL*Plus: Release 12.1.0.2.0 Production on Mon Oct 23 13:35:39 2023

Copyright (c) 1982, 2014, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
```

create_tables.sh

```
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64
"m32cho/@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=
(SID=orcl)))" <<EOF
CREATE TABLE Customer (
   CustomerUserName varchar (30) NOT NULL,
   CustomerName VARCHAR(30),
   CustomerDateOfBirth TIMESTAMP,
   CustomerReserved varchar (30),
   PRIMARY KEY (CustomerUserName));
CREATE TABLE EventHost (
   EventHostUsername VARCHAR(30) NOT NULL,
   EventHostName VARCHAR(30),
   EventHostEventsList VARCHAR(30),
   EventHostContact VARCHAR(200),
   PRIMARY KEY (EventHostUsername));
CREATE TABLE Event(
   EventName VARCHAR(30) NOT NULL,
   EventTime TIMESTAMP,
   EventPrice DECIMAL(10,2),
   EventCapacity INT,
   EventAgeLimit INT,
   PRIMARY KEY (EventName));
CREATE TABLE EventType (
   EventTypeName varchar(30) NOT NULL,
   EventTypeCode varchar(2) NOT NULL,
   Unique (EventTypeCode));
CREATE TABLE EventGenre (
   EventGenreName varchar(30) NOT NULL,
   EventGenrecode varchar(2) NOT NULL,
   Unique (EventGenrecode));
CREATE TABLE EventLocation (
   EventLocationAddress VARCHAR(30) NOT NULL,
   EventLocationCity VARCHAR(30) NOT NULL,
   EventLocationCode VARCHAR(5) NOT NULL,
   Unique (EventLocationAddress));
Create Table Reserve(
   ReserveDate TIMESTAMP,
   CustomerUsername varchar(30) REFERENCES Customer (CustomerUsername),
   EventName varchar(30)REFERENCES Event (EventName),
   primary key (CustomerUsername,EventName) );
CREATE TABLE CreateEvent (
      CreateEventDate TIMESTAMP,
      EventHostUsername VARCHAR(30) REFERENCES EventHost(EventHostUsername),
      EventName VARCHAR(30)  REFERENCES Event(EventName),
      primary key (EventHostUsername,EventName) );
Create Table ContainsType (
      EventName VARCHAR(30) NOT NULL,
      EventTypeCode VARCHAR(5) NOT NULL,
      FOREIGN KEY (EventName) REFERENCES Event(EventName),
      FOREIGN KEY (EventTypeCode) REFERENCES EventType(EventTypeCode));
Create Table ContainsGenre (
      EventName VARCHAR(30) NOT NULL,
      EventGenrecode VARCHAR(5) NOT NULL,
      FOREIGN KEY (EventName) REFERENCES Event(EventName),
```

```
	FOREIGN KEY (EventGenrecode) REFERENCES EventGenre(EventGenrecode));
Create Table ContainsLocation (
	EventName VARCHAR(30) NOT NULL,
	EVENTLOCATIONADDRESS  VARCHAR(30) NOT NULL,
	FOREIGN KEY (EventName) REFERENCES Event(EventName),
	FOREIGN KEY (EVENTLOCATIONADDRESS) REFERENCES EventLocation(EVENTLOCATIONADDRESS));
exit;
EOF
```

```
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> SQL>   2    3    4    5    6
Table created.

SQL> SQL>   2    3    4    5    6
Table created.

SQL> SQL>   2    3    4    5    6    7
Table created.

SQL> SQL>   2    3    4
Table created.

SQL> SQL>   2    3    4
Table created.

SQL> SQL>   2    3    4    5
Table created.

SQL> SQL>   2    3    4    5
Table created.

SQL> SQL>   2    3    4    5
Table created.

SQL> SQL>   2    3    4    5
Table created.

SQL> SQL>   2    3    4    5
Table created.

SQL> SQL>   2    3    4    5
Table created.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
```

populate_tables.sh

```
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64
"m32cho@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF

INSERT INTO EventType (EventTypeName, EventTypeCode) VALUES ('Concert', 'CC');
INSERT INTO EventType VALUES ('Movie', 'MV');
INSERT INTO EventType VALUES ('Sports', 'SP');
INSERT INTO EventType VALUES ('Art', 'AT');

INSERT INTO EventGenre (EventGenreName, EventGenreCode) VALUES ('Basketball', 'BK');
INSERT INTO EventGenre VALUES ('Football', 'FB');
INSERT INTO EventGenre VALUES ('Baseball', 'BB');
INSERT INTO EventGenre VALUES ('Hockey', 'HO');
INSERT INTO EventGenre VALUES ('Action', 'AC');
INSERT INTO EventGenre VALUES ('Adventure', 'AT');
INSERT INTO EventGenre VALUES ('Comedy', 'CD');
INSERT INTO EventGenre VALUES ('Romance', 'RO');
INSERT INTO EventGenre VALUES ('Horror', 'HR');
INSERT INTO EventGenre VALUES ('Mystery', 'MY');
INSERT INTO EventGenre VALUES ('Science-Fiction', 'SF');

INSERT INTO Customer (CustomerUserName, CustomerName, CustomerDateOfBirth, CustomerReserved) VALUES
('john_doe', 'John Doe', TIMESTAMP '2000-03-03 00:00:00', 'Ticket01');

INSERT INTO EventHost (EventHostUsername, EventHostName, EventHostEventsList, EventHostContact) VALUES
('event_host1', 'Event Host 1', 'Kanye Concert', '647-123-4567');

INSERT INTO Event (EventName, EventTime, EventPrice, EventCapacity, EventAgeLimit) VALUES  ('Kanye Concert',
TIMESTAMP '2023-09-30 10:00:00', 50.00, 1000, 19);
INSERT INTO Event VALUES ('Taylor Swift Concert', TIMESTAMP '2023-05-20 10:00:00', 250.00, 2000, 16);
INSERT INTO Event values ('Blue Jays', TIMESTAMP '2023-11-20 10:00:00', 50.00, 2000, 16);
INSERT INTO Event values ('Raptors', TIMESTAMP '2023-12-20 10:00:00', 50.00, 2000, 16);

INSERT INTO EventLocation (EventLocationAddress, EventLocationCity, EventLocationCode) VALUES ('123 Dundas St',
'Toronto', 'T');
INSERT INTO EventLocation VALUES ('1 Dundas St', 'Toronto', 'T');
INSERT INTO EventLocation VALUES ('2 Dundas St', 'Toronto', 'T');
INSERT INTO EventLocation VALUES ('3 Dundas St', 'Toronto', 'T');

INSERT INTO Reserve (ReserveDate, CustomerUsername, EventName) VALUES (TIMESTAMP '2023-09-15 14:30:00',
'john_doe', 'Kanye Concert');
INSERT INTO Reserve VALUES (TIMESTAMP '2023-09-15 14:30:00', 'john_doe', 'Taylor Swift Concert');
INSERT INTO Reserve VALUES (TIMESTAMP '2023-11-20 10:00:00', 'john_doe', 'Blue Jays');
INSERT INTO Reserve VALUES (TIMESTAMP '2023-12-20 10:00:00', 'john_doe', 'Raptors');

INSERT INTO CreateEvent (CREATEEVENTDATE, EVENTHOSTUSERNAME, EVENTNAME) VALUES (TIMESTAMP
'2023-08-20 09:00:00','event_host1','Kanye Concert');
INSERT INTO CreateEvent VALUES (TIMESTAMP '2023-08-20 09:00:00','event_host1','Taylor Swift Concert');
INSERT INTO CreateEvent VALUES (TIMESTAMP '2023-08-20 09:00:00','event_host1','Blue Jays');
INSERT INTO CreateEvent VALUES (TIMESTAMP '2023-08-20 09:00:00','event_host1','Raptors');

INSERT INTO CONTAINSGENRE (EVENTNAME, EVENTGENRECODE) VALUES ('Kanye Concert', 'RO');
INSERT INTO CONTAINSGENRE VALUES ('Taylor Swift Concert', 'HR');
INSERT INTO CONTAINSGENRE VALUES ('Blue Jays', 'BB');
```

```
INSERT INTO CONTAINSGENRE VALUES ('Raptors', 'BK');

INSERT INTO CONTAINSLOCATION (EVENTNAME, EVENTLOCATIONADDRESS) VALUES ('Kanye Concert', '123
Dundas St');
INSERT INTO CONTAINSLOCATION VALUES ('Taylor Swift Concert', '1 Dundas St');
INSERT INTO CONTAINSLOCATION VALUES ('Blue Jays', '2 Dundas St');
INSERT INTO CONTAINSLOCATION VALUES ('Raptors', '3 Dundas St');

INSERT INTO CONTAINSTYPE (EVENTNAME, EVENTTYPECODE) VALUES ('Kanye Concert', 'CC');
INSERT INTO CONTAINSTYPE VALUES ('Taylor Swift Concert', 'CC');
INSERT INTO CONTAINSTYPE VALUES ('Blue Jays', 'SP');
INSERT INTO CONTAINSTYPE VALUES ('Raptors', 'SP');

exit;
EOF
```

```
SQL*Plus: Release 12.1.0.2.0 Production on Mon Oct 23 13:37:10 2023

Copyright (c) 1982, 2014, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL> SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL> SQL>
1 row created.

SQL> SQL>
1 row created.
```

```
SQL>
1 row created.

SQL>
1 row created.

SQL> SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL> SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
```

queries.sh

```
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64 "m32cho/
@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=
orcl)))" <<EOF

SELECT EventName
FROM Event e
WHERE EXISTS (
   SELECT 1
   FROM Reserve r
   WHERE r.EventName = e.EventName
);

SELECT EventType.*, 'Type' AS TypeORGenre
FROM EventType
UNION
SELECT EventGenre.*, 'Genre' AS TypeORGenre
FROM EventGenre;

select eventname
from event e
where  eventcapacity >1000
   MINUS
      (select c.eventname
       from CONTAINSTYPE c
       where c.eventtypecode = 'CC');

SELECT EventName, ReservationCount
FROM (
      SELECT EventName, COUNT(*) AS ReservationCount
      FROM Reserve
      GROUP BY EventName
      ORDER BY ReservationCount DESC
)
WHERE ROWNUM = 1;

SELECT cl.EventLocationAddress, COUNT(cl.EventName) AS EventCount, AVG(e.EventPrice) AS AverageEventPrice
FROM ContainsLocation cl
JOIN Event e ON cl.EventName = e.EventName
GROUP BY cl.EventLocationAddress;
exit;
EOF
```

```
SQL> SQL>   2    3    4    5    6    7
EVENTNAME
------------------------------
Blue Jays
Kanye Concert
Raptors
Taylor Swift Concert

SQL> SQL>   2    3    4    5
EVENTTYPENAME                  EV TYPEO
------------------------------ -- -----
Action                         AC Genre
Adventure                      AT Genre
Art                            AT Type
Baseball                       BB Genre
Basketball                     BK Genre
Comedy                         CD Genre
Concert                        CC Type
Football                       FB Genre
Hockey                         HO Genre
Horror                         HR Genre
Movie                          MV Type

EVENTTYPENAME                  EV TYPEO
------------------------------ -- -----
Mystery                        MY Genre
Romance                        RO Genre
Science-Fiction                SF Genre
Sports                         SP Type

15 rows selected.

SQL> SQL>   2    3    4    5    6    7
EVENTNAME
------------------------------
Blue Jays
Raptors

SQL> SQL>   2    3    4    5    6    7    8
EVENTNAME                      RESERVATIONCOUNT
------------------------------ ----------------
Blue Jays                                     1

SQL> SQL>   2    3    4
EVENTLOCATIONADDRESS           EVENTCOUNT AVERAGEEVENTPRICE
------------------------------ ---------- -----------------
2 Dundas St                             1                50
1 Dundas St                             1               250
3 Dundas St                             1                50
123 Dundas St                           1                50
```

Assignment 6: Functional Dependencies

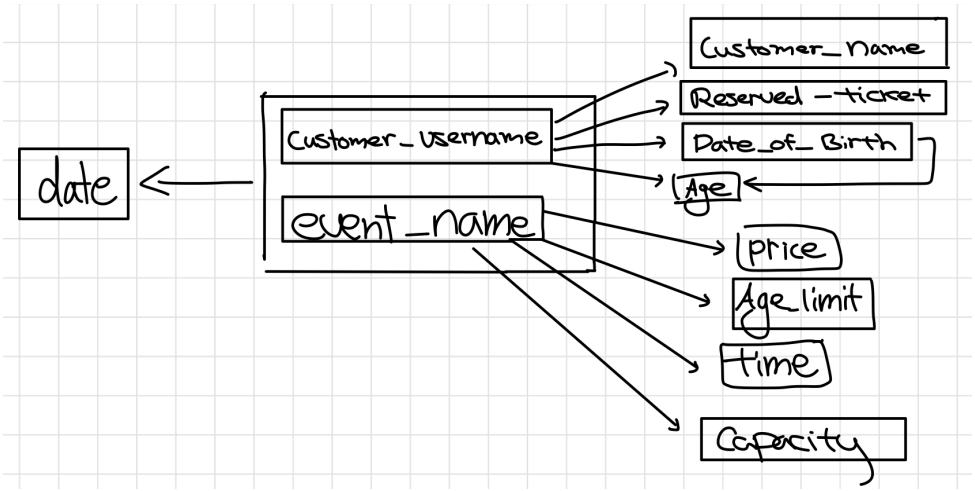
Draft 1

Table A	Relationship	Table B	FN
Customer	Reserve (N-M)	Event	N/A
Host	Create (N-M)	Event	N/A
Event	Contains(1-1)	Type	EventName -> TypeCode TypeCode -> EventName
Event	Contains(1-1)	Genre	EventName -> GenreCode GenreCode -> EventName
Event	Contains(1-1)	Location	EventName -> LocationCode LocationCode -> EventName

Draft 2

Table	Functional Dependency
Customer	CustomerUserName → {CustomerName, CustomerDateOfBirth, CustomerReserved}
EventHost	EventHostUsername → {EventHostName, EventHostEventsList, EventHostContact}
Event	EventName → {EventTime, EventPrice, EventCapacity, EventAgeLimit}
EventType	EventTypeCode → {EventTypeName}
EventGenre	EventGenrecode → {EventGenreName}
EventLocation	EventLocationAddress → {EventLocationCity, EventLocationCode}
Reserve	{CustomerUsername, EventName} → {ReserveDate}
CreateEvent	{EventHostUsername, EventName} → {CreateEventDate}
ContainsType	EventName → {EventTypeCode}
ContainsGenre	EventName → {EventGenreCode}
ContainsLocation	EventName → {EventLocationAddresst}

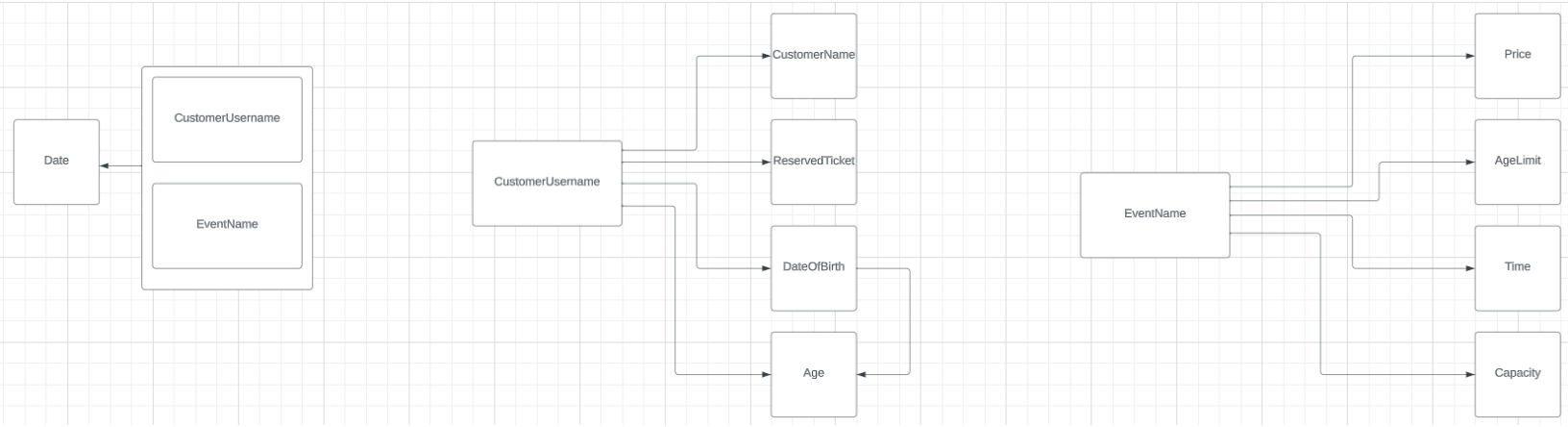
Assignment 7 : 3NF



R = Reserve (<u>cutomer_username, event_name</u>, date, customer_name, reserved_ticket, date_of_birth, age, price, age_limit, time, capacity)

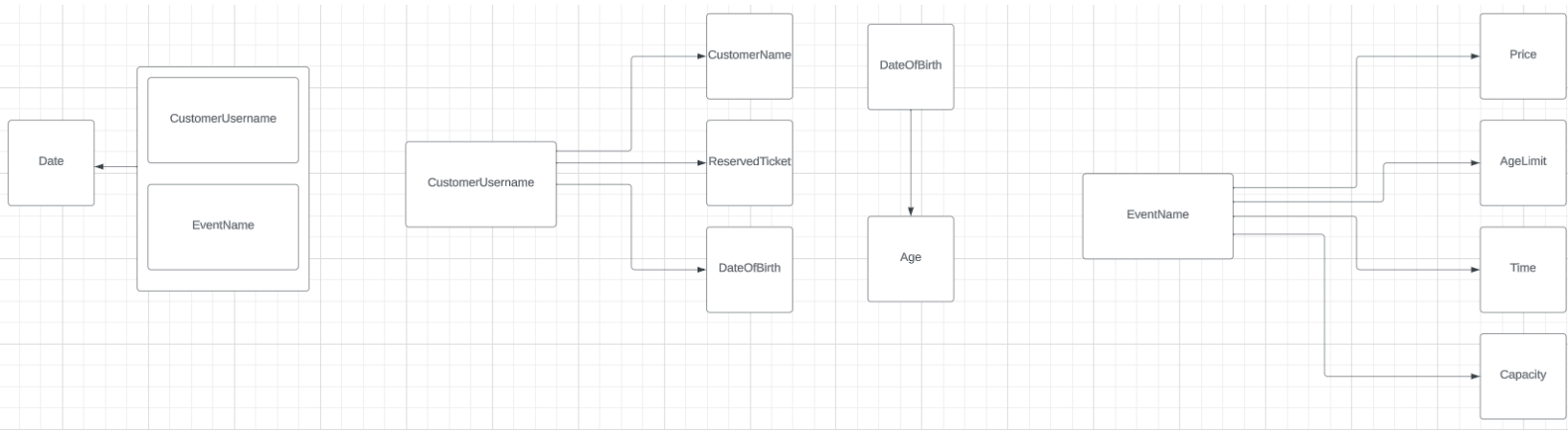
1NF :

1) R1 : customer_username, event_name → date, customer_name, reserved_ticket, date_of_birth, age, price, age_limit, time, capacity
2) R2.1 : Customer_username → customer_name
   R2.2 : Customer_username → reserved_ticket
   R2.3 : Customer_username → date_of_birth
   R2.4 : Customer_username → age
3) R3.1 : Event_name → price
   R3.2 : Event_name → age_limit
   R3.3 : Event_name → time
   R3.4 : Event_name → capacity



2NF :

1) R1 (<u>customer_username, event_name</u>, date)
   FD : customer_username, event_name → date
2) R2 (<u>customer_username</u>, customer_name, reserved_ticket, date_of_birth, age)
   FD : Customer_username → customer_name, reserved_ticket, date_of_birth, age
   Date_of_birth → age
3) R3 (<u>event_name</u>, price, age_limit, time, capacity)
   FD : event_name -> price, age_limit, time, capacity



3NF:

1) R1 (<u>customer_username, event_name</u>, date)
   FD : customer_username, event_name → date
2) R2.1 (<u>customer_username</u>, customer_name, reserved_ticket, date_of_birth
   FD : Customer_username → customer_name, reserved_ticket, date_of_birth
   R2.2 (date_of_birth, age)
   FD : Date_of_birth → age
3) R3 (<u>event_name</u>, price, age_limit, time, capacity)
   FD : event_name -> price, age_limit, time, capacity

Customer Table

R = Customer (<u>cutomer_username</u>, customer_name, customer_date_of_birth, cusomter_reserved)

1NF:

- R1: (<u>cutomer_username</u>, customer_name, customer_date_of_birth, cusomter_reserved, age)

2NF:

- R1: (<u>cutomer_username</u>, customer_name, customer_date_of_birth, cusomter_reserved, age)
  FD: cutomer_username → customer_name, cusomter_reserved
  customer_date_of_birth → age

3NF:

- R1.1: (<u>cutomer_username</u>, customer_name, customer_date_of_birth, cusomter_reserved)
  FD: cutomer_username → customer_name, cusomter_reserved
- R1.2: (customer_date_of_birth, age)
  FD: customer_date_of_birth → age

EventHost Table

R = EventHost (<u>EventHostUsername</u>, EventHostName, EventHostEventsList, EventHostContact)

1NF:

1) R1: (<u>EventHostUsername</u>, EventHostName, EventHostEventsList, EventHostContact)

2NF:

1) R1: (<u>EventHostUsername</u>, EventHostEventsList, EventHostContact)
   FD: EventHostUsername → EventHostEventsList, EventHostContact
2) R2: (<u>EventHostUsername</u>, EventHostName)
   FD: EventHostUsername → EventHostName

3NF:

1) R1: (<u>EventHostUsername</u>, EventHostEventsList, EventHostContact)
   FD: EventHostUsername → EventHostEventsList, EventHostContact
2) R2: (<u>EventHostUsername</u>, EventHostName)
   FD: EventHostUsername → EventHostName

Event Table

R = Event (<u>EventName</u>, EventTime, EventPrice, EventCapacity, EventAgeLimit)

1) R1: (<u>EventName</u>, EventTime, EventPrice, EventCapacity, EventAgeLimit)
   FD: EventName → EventTime, EventPrice, EventCapacity, EventAgeLimit

Already in 3NF as all non-key attributes depend directly on the primary key (EventName) and there are no transitive dependencies.

EventType Table

R = EventType (<u>EventTypeName</u>, EventTypeCode)

1) R1: (<u>EventTypeCode</u>, EventTypeName)
   FD: EventTypeCode → EventTypeName

Already in 3NF as all non-key attributes depend directly on the primary key (EventTypeCode) and there are no transitive dependencies.

EventGenre Table

R = EventGenre (EventGenreName, EventGenreCode)

1) R1: (EventGenreCode, EventGenreName)
   FD: EventGenreCode → EventGenreName

Already in 3NF as all non-key attributes depend directly on the primary key (EventTypeCode) and there are no transitive dependencies.

EventLocation Table

R = EventLocation (EventLocationAddress, EventLocationCity, EventLocationCode)

1) R1: (EventLocationAddress, EventLocationCity, EventLocationCode)
   FD: EventLocationAddress → EventLocationCity, EventLocationCode

Already in 3NF as all non-key attributes depend directly on the primary key (EventTypeCode) and there are no transitive dependencies.

CreateEvent Table

R = CreateEvent (EventHostUsername, EventName, date, EventHostName, EventHostEventsList, EventHostContact, price, age_limit, time, capacity)

1NF :

1) R1 : EventHostUsername, EventName → date, EventHostName, EventHostEventsList, EventHostContact, price, age_limit, time, capacity
2) R2.1 : EventHostUsername → EventHostName
   R2.2 : EventHostUsername → EventHostEventsList
   R2.3 : EventHostUsername → EventHostContact
3) R3.1 : Event_name → price
   R3.2 : Event_name → age_limit
   R3.3 : Event_name → time
   R3.4 : Event_name → capacity

2NF :

1) R1: (EventHostUsername, EventName, date)
   FD : EventHostUsername, EventName → date
2) R2 (EventHostUsername, EventHostName, EventHostEventsList, EventHostContact)
   FD : EventHostUsername → EventHostName, EventHostEventsList, EventHostContact
3) R3: (EventName, price, age_limit, time, capacity)
   FD : EventName -> price, age_limit, time, capacity

3NF:

1) R1 (EventHostUsername, EventName, date)
   FD : EventHostUsername, EventName → date
2) R2: (EventHostUsername, EventHostName, EventHostEventsList, EventHostContact
   FD: EventHostUsername → EventHostName, EventHostEventsList, EventHostContact
3) R3: (EventName, price, age_limit, time, capacity)
   FD : EventName -> price, age_limit, time, capacity

Transitive Functional Dependency:

EVENTLOCATION:

EventLocationAddress → {EventLocationCity, EventLocationCode}
EventLocationCity → {EventLocationCode}

Compound Primary Key:

CREATEEVENT:

{EventHostUsername, EventName} → {CreateEventDate}

Is a partial dependency because

EventName → {CreateEventDate} & EventHostUsername -> {CreateEventDate}

holds.

Assignment 8 : BCNF

Customer Table

R = Customer (cutomer_username, customer_name, customer_date_of_birth, cusomter_reserved)

R1 - FD: customer_username → customer_name, customer_reserved

customer_username$^+$ = {customer_username, customer_name, customer_reserved}

BCNF since there is only one Functional Dependency and it is a superkey.

R2 - FD: customer_date_of_birth → age

customer_date_of_birth$^+$ = {customer_date_of_birth, age}

BCNF since there is only one Functional Dependency and it is a superkey.

EventHost Table

R = EventHost (EventHostUsername, EventHostName, EventHostEventsList, EventHostContact)

FD: EventHostUsername → EventHostEventsList, EventHostContact, EventHostName

Event Table

R = Event (EventName, EventTime, EventPrice, EventCapacity, EventAgeLimit)

FD: EventName → EventTime, EventPrice, EventCapacity, EventAgeLimit

EventName$^+$ = {EventName, EventTime, EventPrice, EventCapacity, EventAgeLimit}

BCNF since there is only one Functional Dependency and it is a superkey.

EventType Table

R = EventType (EventTypeName, EventTypeCode)

1) R1: (EventTypeCode, EventTypeName)
   FD: EventTypeCode → EventTypeName

Is in BCNF because it is already in 3NF, and the functional dependency A → B holds, where A (EventTypeCode) is the superkey.

EventGenre Table

R = EventGenre (EventGenreName, EventGenreCode)

1) R1: (EventGenreCode, EventGenreName)
   FD: EventGenreCode → EventGenreName

Is in BCNF because it is already in 3NF, and the functional dependency A → B holds, where A (EventGenreCode) is the superkey.

EventLocation Table

R = EventLocation (EventLocationAddress, EventLocationCity, EventLocationCode)

1) R1: (EventLocationAddress, EventLocationCity, EventLocationCode)
   FD: EventLocationAddress → EventLocationCity, EventLocationCode

Is in BCNF because it is already in 3NF, and the functional dependency A → B, C holds, where A (EventLocationAddress) is the superkey.

CreateEvent Table

R = CreateEvent (EventHostUsername, EventName, date, EventHostName, EventHostEventsList, EventHostContact, price, age_limit, time, capacity)

FD : EventHostUsername, EventName → date

EventHostUsername → EventHostName, EventHostEventsList, EventHostContact

EventName -> price, age_limit, time, capacity

All of the left hand sides are the primary keys and right hand sides are the non-candidate.

Reserve Table

R = Reserve (cutomer_username, event_name, date, customer_name, reserved_ticket, date_of_birth, age, price, age_limit, time, capacity)

FD : customer_username, event_name → date

customer_username → customer_name, reserved_ticket, date_of_birth, age

event_name -> price, age_limit, time, capacity

All of the left hand sides are the primary keys and right hand sides are the non-candidate.

Assignment 9 : Alter table

```
m32cho@europa:~$ bash alter.sh
=================================================================
 Which table would you like to alter? Please Type it in
 Event
 Customer
 Reserve
 EventType
 EventLocation
 EventHost
 EventGenre
 CreateEvent
 ContainsLocation
 ContainsGenre
 ContainsType
Event
```

Adding Column

```
How would you like to alter?
  1) Adding Column
  2) Modifying Column
Choose:
1
Type the name of the column:
Name
Type the type of the column:
varchar(20)

SQL*Plus: Release 12.1.0.2.0 Production on Sun Nov 26 21:40:46 2023

Copyright (c) 1982, 2014, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL>   2
Table altered.
```

Modifying Column

```
How would you like to alter?
  1) Adding Column
  2) Modifying Column
Choose:
2
Type the name of the column:
Name
Type the type of the column:
char(2)

SQL*Plus: Release 12.1.0.2.0 Production on Sun Nov 26 21:41:48 2023

Copyright (c) 1982, 2014, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL>   2
Table altered.
```

Alter.sh

```
#!/bin/sh
echo "==================================================================="
echo " Which table would you like to alter? Please Type it in"
echo " Event"
echo " Customer"
echo " Reserve"
echo " EventType"
echo " EventLocation"
echo " EventHost"
echo " EventGenre"
echo " CreateEvent"
echo " ContainsLocation"
echo " ContainsGenre"
echo " ContainsType"
read tableName
echo "How would you like to alter?"
echo " $IS_SELECTED1 1) Adding Column"
echo " $IS_SELECTED2 2) Modifying Column"
echo "Choose: "
read CHOICE
if [ "$CHOICE" == "1" ]
	then
	echo "Type the name of the column: "
	read addColumn
	echo "Type the type of the column: "
	read addType
	export tableName
	export addColumn
	export addType
	bash add_column.sh
	exit;
elif [ "$CHOICE" == "2" ]
	then
	echo "Type the name of the column: "
	read modifyColumn
	echo "Type the type of the column: "
	read modifyType
	export tableName
	export modifyColumn
	export modifyType
	bash modify_column.sh
	exit;
fi
```

Add_column.sh

```
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64
"m32cho/01195255@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Por
t=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF
ALTER TABLE $tableName ADD(
	$addColumn $addType );
exit;
EOF
```

Modify_column.sh

```
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64
"m32cho/01195255@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.sc
s.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF
ALTER TABLE $tableName MODIFY(
	$modifyColumn $modifyType );
exit;
EOF
```

Assignment 10 : Relational Algebra

1) Selection : List all the attributes of events with the age limit of 19

$\sigma_{ageLimit >= 19}(Event)$

2) Projection: List event name and event price of events with the age limit of 19

$\pi_{eventName,\ eventPrice}(\sigma_{ageLimit >= 19}(Event))$

3) Differences: Give a list of all names of events with the age limit of 19, but the price is less than 100.

A = events with the age limit of 19
B = events with the price bigger than 100
A = $\pi_{eventName}\ (\sigma_{ageLimit >= 19}(Event))$
B = $\pi_{eventName}\ (\sigma_{eventPrice>=100}(Event))$
Result = A - B

4) Union : Give a list of all names of events that are not reserved by ‘John’ or not created by ‘Tim’.

A = not reserved for ‘John’ = $\pi_{eventName}$ (events) - $\pi_{eventName}\ (\sigma_{customerName = \text{‘John’}}(Customer))$
B = not created by ‘Tim’ = $\pi_{eventName}$ (events) - $\pi_{eventName}\ (\sigma_{hostName = \text{‘Tim’}}(EventHost))$
Result = A U B

5) List all events which title is not “Finding Nemo”

$\sigma_{NOT(ageLimit = 19)}\ (Event)$

6) Theta-Join (Cartesian) : List event name and event price of events in Concert type

$\pi_{eventName,\ eventPrice}(\sigma_{EventTypeCode = \text{‘CC’}}(containsType \mid\mid EventType))$

GUI

Index.php

```
<?php
error_reporting(E_ALL);
ini_set('display_errors', 'On');

$conn = oci_connect('z86khan',
'03228606','(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.c
a)(Port=1521))(CONNECT_DATA=(SID=orcl)))');

if (!$conn) {
    $m = oci_error();
    echo "<div class='error'>Error connecting to the Oracle database: " .
$m['message'] . "</div>";
} else {
    echo "<div class='success'>Successfully connected to the Oracle
database!</div>";
}

    // Handle user actions
    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        if (isset($_POST['action'])) {
            $action = $_POST['action'];

            var_dump($_POST);

            switch ($action) {
                case 'create':
                    createTables($conn);
                    echo "Tables created successfully!";
                    break;
                case 'drop':
                    dropTables($conn);
                    echo "Tables dropped successfully!";
                    break;
                case 'populate':
                    populateTables($conn);
                    echo "Tables populated successfully!";
                    break;
```

```
                case 'query':
                    executeQueries($conn);
                    echo "Tables queried successfully!";
                    break;
                case 'createEvent':
                    $eventName = $_POST['eventName'];
                    $eventTime = $_POST['eventTime'];
                    $eventPrice = $_POST['eventPrice'];
                    $eventCapacity = $_POST['eventCapacity'];
                    $eventAgeLimit = $_POST['eventAgeLimit'];

                    createEvent($conn, $eventName, $eventTime,
$eventPrice, $eventCapacity, $eventAgeLimit);
                    echo "Event created successfully!";
                    break;
                default:
                    echo "Invalid action!";
            }
        }
    }


function executeQueries($conn) {
    // Query 1
    $query1 = "SELECT EventName
    FROM Event e
    WHERE EXISTS (
        SELECT 1
        FROM Reserve r
        WHERE r.EventName = e.EventName
    )";
    $result1 = oci_parse($conn, $query1);
    oci_execute($result1);
    echo "<h2>Query 1: Events with Reservations</h2>";
    echo "<table border='1'>";
    echo "<tr><th>Event Name</th></tr>";
    while ($row = oci_fetch_assoc($result1)) {
        echo "<tr><td>" . $row['EVENTNAME'] . "</td></tr>";
    }
    echo "</table>";
```

```php
// Query 2
$query2 = "SELECT EventType.*, 'Type' AS TypeORGenre
FROM EventType
UNION
SELECT EventGenre.*, 'Genre' AS TypeORGenre
FROM EventGenre";
$result2 = oci_parse($conn, $query2);
oci_execute($result2);
echo "<h2>Query 2: Event Types and Genres</h2>";
echo "<table border='1'>";
echo "<tr><th>Event Type/Genre</th></tr>";
while ($row = oci_fetch_assoc($result2)) {
    echo "<tr><td>" . print_r($row, true) . "</td></tr>";
}
echo "</table>";

// Query 3
$query3 = "SELECT EventName
FROM Event e
WHERE EventCapacity > 1000
MINUS
(SELECT c.EventName
FROM CONTAINSTYPE c
WHERE c.EventTypeCode = 'CC')";
$result3 = oci_parse($conn, $query3);
oci_execute($result3);
echo "<h2>Query 3: High Capacity Events excluding Concerts</h2>";
echo "<table border='1'>";
echo "<tr><th>Event Name</th></tr>";
while ($row = oci_fetch_assoc($result3)) {
    echo "<tr><td>" . $row['EVENTNAME'] . "</td></tr>";
}
echo "</table>";

// Query 4
$query4 = "SELECT EventName, ReservationCount
FROM (
    SELECT EventName, COUNT(*) AS ReservationCount
    FROM Reserve
```

```
        GROUP BY EventName
        ORDER BY ReservationCount DESC
    )
    WHERE ROWNUM = 1";
    $result4 = oci_parse($conn, $query4);
    oci_execute($result4);
    echo "<h2>Query 4: Event with the Most Reservations</h2>";
    echo "<table border='1'>";
    echo "<tr><th>Event Name</th><th>Reservation Count</th></tr>";
    while ($row = oci_fetch_assoc($result4)) {
        echo "<tr><td>" . $row['EVENTNAME'] . "</td><td>" .
$row['RESERVATIONCOUNT'] . "</td></tr>";
    }
    echo "</table>";

    // Query 5
    $query5 = "SELECT cl.EventLocationAddress, COUNT(cl.EventName) AS
EventCount, AVG(e.EventPrice) AS AverageEventPrice
    FROM ContainsLocation cl
    JOIN Event e ON cl.EventName = e.EventName
    GROUP BY cl.EventLocationAddress";
    $result5 = oci_parse($conn, $query5);
    oci_execute($result5);
    echo "<h2>Query 5: Event Counts and Average Price by Location</h2>";
    echo "<table border='1'>";
    echo "<tr><th>Location</th><th>Event Count</th><th>Average Event
Price</th></tr>";
    while ($row = oci_fetch_assoc($result5)) {
        echo "<tr><td>" . $row['EVENTLOCATIONADDRESS'] . "</td><td>" .
$row['EVENTCOUNT'] . "</td><td>" . $row['AVERAGEEVENTPRICE'] .
"</td></tr>";
    }
    echo "</table>";

    // Query 6
    $query6 = "SELECT * FROM Event";
    $result6 = oci_parse($conn, $query6);
    oci_execute($result6);
    echo "<h2>Query 6: Events and Their Properties</h2>";
    echo "<table border='1'>";
```

```
    echo "<tr><th>Event Name</th><th>Event Time</th><th>Event
Price</th><th>Event Capacity</th><th>Event Age Limit</th></tr>";

    while ($row = oci_fetch_assoc($result6)) {
    echo "<tr>";
    echo "<td>" . $row['EVENTNAME'] . "</td>";
    echo "<td>" . $row['EVENTTIME'] . "</td>";
    echo "<td>" . $row['EVENTPRICE'] . "</td>";
    echo "<td>" . $row['EVENTCAPACITY'] . "</td>";
    echo "<td>" . $row['EVENTAGELIMIT'] . "</td>";
    echo "</tr>";
    }
    echo "</table>";
}

function dropTables($conn) {
    $tablesToDrop = [
        'CONTAINSTYPE',
        'CONTAINSGENRE',
        'CONTAINSLOCATION',
        'CREATEEVENT',
        'EVENTGENRE',
        'EVENTHOST',
        'EVENTLOCATION',
        'EVENTTYPE',
        'RESERVE',
        'CUSTOMER',
        'EVENT'
    ];

    foreach ($tablesToDrop as $table) {
        $query = "DROP TABLE $table CASCADE CONSTRAINTS";
        $stmt = oci_parse($conn, $query);
        oci_execute($stmt);
    }
}

function createTables($conn) {
    $queries = [
        "CREATE TABLE Customer (
```

```
            CustomerUserName varchar (30) NOT NULL,
            CustomerName VARCHAR(30),
            CustomerDateOfBirth TIMESTAMP,
            CustomerReserved varchar (30),
            PRIMARY KEY (CustomerUserName))",
        "CREATE TABLE EventHost (
            EventHostUsername VARCHAR(30) NOT NULL,
            EventHostName VARCHAR(30),
            EventHostEventsList VARCHAR(30),
            EventHostContact VARCHAR(200),
            PRIMARY KEY (EventHostUsername))",
        "CREATE TABLE Event (
            EventName VARCHAR(30) NOT NULL,
            EventTime TIMESTAMP,
            EventPrice DECIMAL(10,2),
            EventCapacity INT,
            EventAgeLimit INT,
            PRIMARY KEY (EventName))",
        "CREATE TABLE EventType (
            EventTypeName varchar(30) NOT NULL,
            EventTypeCode varchar(2) NOT NULL,
            UNIQUE (EventTypeCode))",
        "CREATE TABLE EventGenre (
            EventGenreName varchar(30) NOT NULL,
            EventGenrecode varchar(2) NOT NULL,
            UNIQUE (EventGenrecode))",
        "CREATE TABLE EventLocation (
            EventLocationAddress VARCHAR(30) NOT NULL,
            EventLocationCity VARCHAR(30) NOT NULL,
            EventLocationCode VARCHAR(5) NOT NULL,
            UNIQUE (EventLocationAddress))",
        "CREATE TABLE Reserve (
            ReserveDate TIMESTAMP,
            CustomerUsername varchar(30),
            EventName varchar(30),
            PRIMARY KEY (CustomerUsername, EventName),
            FOREIGN KEY (CustomerUsername) REFERENCES
Customer(CustomerUserName),
            FOREIGN KEY (EventName) REFERENCES Event(EventName))",
        "CREATE TABLE CreateEvent (
```

```
            CreateEventDate TIMESTAMP,
            EventHostUsername VARCHAR(30),
            EventName VARCHAR(30),
            PRIMARY KEY (EventHostUsername, EventName),
            FOREIGN KEY (EventHostUsername) REFERENCES
EventHost(EventHostUsername),
            FOREIGN KEY (EventName) REFERENCES Event(EventName))",
        "CREATE TABLE ContainsType (
            EventName VARCHAR(30),
            EventTypeCode VARCHAR(5),
            FOREIGN KEY (EventName) REFERENCES Event(EventName),
            FOREIGN KEY (EventTypeCode) REFERENCES
EventType(EventTypeCode))",
        "CREATE TABLE ContainsGenre (
            EventName VARCHAR(30),
            EventGenrecode VARCHAR(5),
            FOREIGN KEY (EventName) REFERENCES Event(EventName),
            FOREIGN KEY (EventGenrecode) REFERENCES
EventGenre(EventGenrecode))",
        "CREATE TABLE ContainsLocation (
            EventName VARCHAR(30),
            EVENTLOCATIONADDRESS  VARCHAR(30),
            FOREIGN KEY (EventName) REFERENCES Event(EventName),
            FOREIGN KEY (EVENTLOCATIONADDRESS) REFERENCES
EventLocation(EVENTLOCATIONADDRESS))"
    ];

    foreach ($queries as $query) {
        $stmt = oci_parse($conn, $query);
        oci_execute($stmt);
    }
}

function populateTables($conn) {
    $queries = [
        "INSERT INTO EventType (EventTypeName, EventTypeCode) VALUES
('Concert', 'CC')",
        "INSERT INTO EventType VALUES ('Movie', 'MV')",
        "INSERT INTO EventType VALUES ('Sports', 'SP')",
        "INSERT INTO EventType VALUES ('Art', 'AT')",
```

```
        "INSERT INTO EventGenre (EventGenreName, EventGenreCode) VALUES
('Basketball', 'BK')",
        "INSERT INTO EventGenre VALUES ('Football', 'FB')",
        "INSERT INTO EventGenre VALUES ('Baseball', 'BB')",
        "INSERT INTO EventGenre VALUES ('Hockey', 'HO')",
        "INSERT INTO EventGenre VALUES ('Action', 'AC')",
        "INSERT INTO EventGenre VALUES ('Adventure', 'AT')",
        "INSERT INTO EventGenre VALUES ('Comedy', 'CD')",
        "INSERT INTO EventGenre VALUES ('Romance', 'RO')",
        "INSERT INTO EventGenre VALUES ('Horror', 'HR')",
        "INSERT INTO EventGenre VALUES ('Mystery', 'MY')",
        "INSERT INTO EventGenre VALUES ('Science-Fiction', 'SF')",


        "INSERT INTO Customer (CustomerUserName, CustomerName,
CustomerDateOfBirth, CustomerReserved) VALUES ('john_doe', 'John Doe',
TIMESTAMP '2000-03-03 00:00:00', 'Ticket01')",


        "INSERT INTO EventHost (EventHostUsername, EventHostName,
EventHostEventsList, EventHostContact) VALUES ('event_host1', 'Event Host
1', 'Kanye Concert', '647-123-4567')",


        "INSERT INTO Event (EventName, EventTime, EventPrice,
EventCapacity, EventAgeLimit) VALUES ('Kanye Concert', TIMESTAMP
'2023-09-30 10:00:00', 50.00, 1000, 19)",
        "INSERT INTO Event VALUES ('Taylor Swift Concert', TIMESTAMP
'2023-05-20 10:00:00', 250.00, 2000, 16)",
        "INSERT INTO Event VALUES ('Blue Jays', TIMESTAMP '2023-11-20
10:00:00', 50.00, 2000, 16)",
        "INSERT INTO Event VALUES ('Raptors', TIMESTAMP '2023-12-20
10:00:00', 50.00, 2000, 16)",


        "INSERT INTO EventLocation (EventLocationAddress,
EventLocationCity, EventLocationCode) VALUES ('123 Dundas St', 'Toronto',
'T')",
```

```
        "INSERT INTO EventLocation VALUES ('1 Dundas St', 'Toronto',
'T')",
        "INSERT INTO EventLocation VALUES ('2 Dundas St', 'Toronto',
'T')",
        "INSERT INTO EventLocation VALUES ('3 Dundas St', 'Toronto',
'T')",


        "INSERT INTO Reserve (ReserveDate, CustomerUsername, EventName)
VALUES (TIMESTAMP '2023-09-15 14:30:00', 'john_doe', 'Kanye Concert')",
        "INSERT INTO Reserve VALUES (TIMESTAMP '2023-09-15 14:30:00',
'john_doe', 'Taylor Swift Concert')",
        "INSERT INTO Reserve VALUES (TIMESTAMP '2023-11-20 10:00:00',
'john_doe', 'Blue Jays')",
        "INSERT INTO Reserve VALUES (TIMESTAMP '2023-12-20 10:00:00',
'john_doe', 'Raptors')",


        "INSERT INTO CreateEvent (CREATEEVENTDATE, EVENTHOSTUSERNAME,
EVENTNAME) VALUES (TIMESTAMP '2023-08-20 09:00:00','event_host1','Kanye
Concert')",
        "INSERT INTO CreateEvent VALUES (TIMESTAMP '2023-08-20
09:00:00','event_host1','Taylor Swift Concert')",
        "INSERT INTO CreateEvent VALUES (TIMESTAMP '2023-08-20
09:00:00','event_host1','Blue Jays')",
        "INSERT INTO CreateEvent VALUES (TIMESTAMP '2023-08-20
09:00:00','event_host1','Raptors')",


        "INSERT INTO CONTAINSGENRE (EVENTNAME, EVENTGENRECODE) VALUES
('Kanye Concert', 'RO')",
        "INSERT INTO CONTAINSGENRE VALUES ('Taylor Swift Concert', 'HR')",
        "INSERT INTO CONTAINSGENRE VALUES ('Blue Jays', 'BB')",
        "INSERT INTO CONTAINSGENRE VALUES ('Raptors', 'BK')",


        "INSERT INTO CONTAINSLOCATION (EVENTNAME, EVENTLOCATIONADDRESS)
VALUES ('Kanye Concert', '123 Dundas St')",
        "INSERT INTO CONTAINSLOCATION VALUES ('Taylor Swift Concert', '1
Dundas St')",
```

```
        "INSERT INTO CONTAINSLOCATION VALUES ('Blue Jays', '2 Dundas
St')",
        "INSERT INTO CONTAINSLOCATION VALUES ('Raptors', '3 Dundas St')",


        "INSERT INTO CONTAINSTYPE (EVENTNAME, EVENTTYPECODE) VALUES
('Kanye Concert', 'CC')",
        "INSERT INTO CONTAINSTYPE VALUES ('Taylor Swift Concert', 'CC')",
        "INSERT INTO CONTAINSTYPE VALUES ('Blue Jays', 'SP')",
        "INSERT INTO CONTAINSTYPE VALUES ('Raptors', 'SP')"
        ];

    foreach ($queries as $query) {
        $stmt = oci_parse($conn, $query);
        oci_execute($stmt);
    }
}

function createEvent($conn, $eventName, $eventTime, $eventPrice,
$eventCapacity, $eventAgeLimit) {
    $query = "INSERT INTO Event (EventName, EventTime, EventPrice,
EventCapacity, EventAgeLimit)
               VALUES (:eventName, TO_TIMESTAMP(:eventTime,
'YYYY-MM-DD\"T\"HH24:MI'), :eventPrice, :eventCapacity, :eventAgeLimit)";
    $stmt = oci_parse($conn, $query);

    $eventTime = date('Y-m-d\TH:i', strtotime($eventTime));

    oci_bind_by_name($stmt, ":eventName", $eventName);
    oci_bind_by_name($stmt, ":eventTime", $eventTime);
    oci_bind_by_name($stmt, ":eventPrice", $eventPrice, PDO::PARAM_STR);
    oci_bind_by_name($stmt, ":eventCapacity", $eventCapacity);
    oci_bind_by_name($stmt, ":eventAgeLimit", $eventAgeLimit);

    if (oci_execute($stmt)) {
        echo "Event created successfully!";
    } else {
        $error = oci_error($stmt);
        echo "Error creating event: " . $error['message'];
    }
```

```
}



?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Database Management</title>
    <style>
        body {
            font-family: 'Roboto', sans-serif;
            background-color: #333333;
            margin: 0;
            padding: 20px;
            color: #fff;
        }

        h1 {
            text-align: center;
        }

        form {
            background-color: #fff;
            padding: 20px;
            border-radius: 5px;
            max-width: 400px;
            margin: 20px auto;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            color: #333;
        }

        label {
            display: block;
            margin-bottom: 8px;
        }
```

```
        select, input[type="text"], input[type="datetime-local"],
input[type="number"] {
            width: 100%;
            padding: 10px;
            margin-bottom: 20px;
            box-sizing: border-box;
        }

        input[type="submit"] {
            background-color: #4caf50;
            color: #fff;
            padding: 10px 20px;
            border: none;
            border-radius: 3px;
            cursor: pointer;
        }

        input[type="submit"]:hover {
            background-color: #45a049;
        }

        h2 {
            margin-top: 30px;
            color: #fff;
        }

        pre {
            background-color: #f8f8f8;
            padding: 10px;
            border: 1px solid #ddd;
            border-radius: 5px;
            overflow: auto;
            color: #333;
        }

        table {
            width: 100%;
            border-collapse: collapse;
            margin-top: 20px;
            color: #fff;
```

```
        }

        th, td {
            padding: 10px;
            text-align: left;
            border-bottom: 1px solid #ddd;
        }

        th {
            background-color: #4caf50;
            color: #fff;
        }

        .success {
            color: #4caf50;
        }

        .error {
            color: #f44336;
        }
    </style>
</head>
<body>
    <center><h1>E-Ticket Reservation System</h1></center>

    <form method="post" action="">
        <label for="action">Select Action:</label>
        <select name="action" id="action">
            <option value="create">Create Tables</option>
            <option value="drop">Drop Tables</option>
            <option value="populate">Populate Tables</option>
            <option value="query">Query Tables</option>
        </select>
        <br>

        <input type="submit" value="Submit">
    </form>

    <form method="post" action="">
        <label for="eventName">Event Name:</label>
```

```
        <input type="text" name="eventName" id="eventName" required>
        <br>
        <label for="eventTime">Event Time:</label>
        <input type="datetime-local" name="eventTime" id="eventTime"
required>
        <br>
        <label for="eventPrice">Event Price:</label>
        <input type="number" name="eventPrice" id="eventPrice" step="0.01"
required>
        <br>
        <label for="eventCapacity">Event Capacity:</label>
        <input type="number" name="eventCapacity" id="eventCapacity"
required>
        <br>
        <label for="eventAgeLimit">Event Age Limit:</label>
        <input type="number" name="eventAgeLimit" id="eventAgeLimit"
required>
        <br>

        <input type="hidden" name="action" value="createEvent">
        <input type="submit" value="Submit">
    </form>

</body>
</html>
```

Database Management

Select Action: Create Tables

Submit

Create Tables
Drop Tables
Populate Tables
Query Tables

Query 1:

```
Blue Jays
Kanye Concert
Raptors
Taylor Swift Concert
```

Query 2:

```
Array
(
    [EVENTTYPENAME] => Action
    [EVENTTYPECODE] => AC
    [TYPEORGENRE] => Genre
)

Array
(
    [EVENTTYPENAME] => Adventure
    [EVENTTYPECODE] => AT
    [TYPEORGENRE] => Genre
)

Array
(
    [EVENTTYPENAME] => Art
    [EVENTTYPECODE] => AT
    [TYPEORGENRE] => Type
)

Array
(
    [EVENTTYPENAME] => Baseball
    [EVENTTYPECODE] => BB
    [TYPEORGENRE] => Genre
)

Array
(
    [EVENTTYPENAME] => Basketball
    [EVENTTYPECODE] => BK
    [TYPEORGENRE] => Genre
)

Array
(
    [EVENTTYPENAME] => Comedy
    [EVENTTYPECODE] => CD
    [TYPEORGENRE] => Genre
)

Array
(
    [EVENTTYPENAME] => Concert
    [EVENTTYPECODE] => CC
    [TYPEORGENRE] => Type
)
```

Query 3:

```
Blue Jays
Raptors
```

Query 4:

```
Event Name: Blue Jays, Reservation Count: 1
```

Query 5:

```
Location: 2 Dundas St, Event Count: 2, Average Event Price: 50
Location: 1 Dundas St, Event Count: 2, Average Event Price: 250
Location: 3 Dundas St, Event Count: 2, Average Event Price: 50
Location: 123 Dundas St, Event Count: 2, Average Event Price: 50
```

Tables queried successfully!