Convention Centre DBMS

Group 13

Stephanie Huynh	500953471
Laith Kamal	500946949
Jenny Su	500962385
Adam Whittington	500912411

CPS510: Database Systems I

Dr. A. Abhari

December 2, 2021

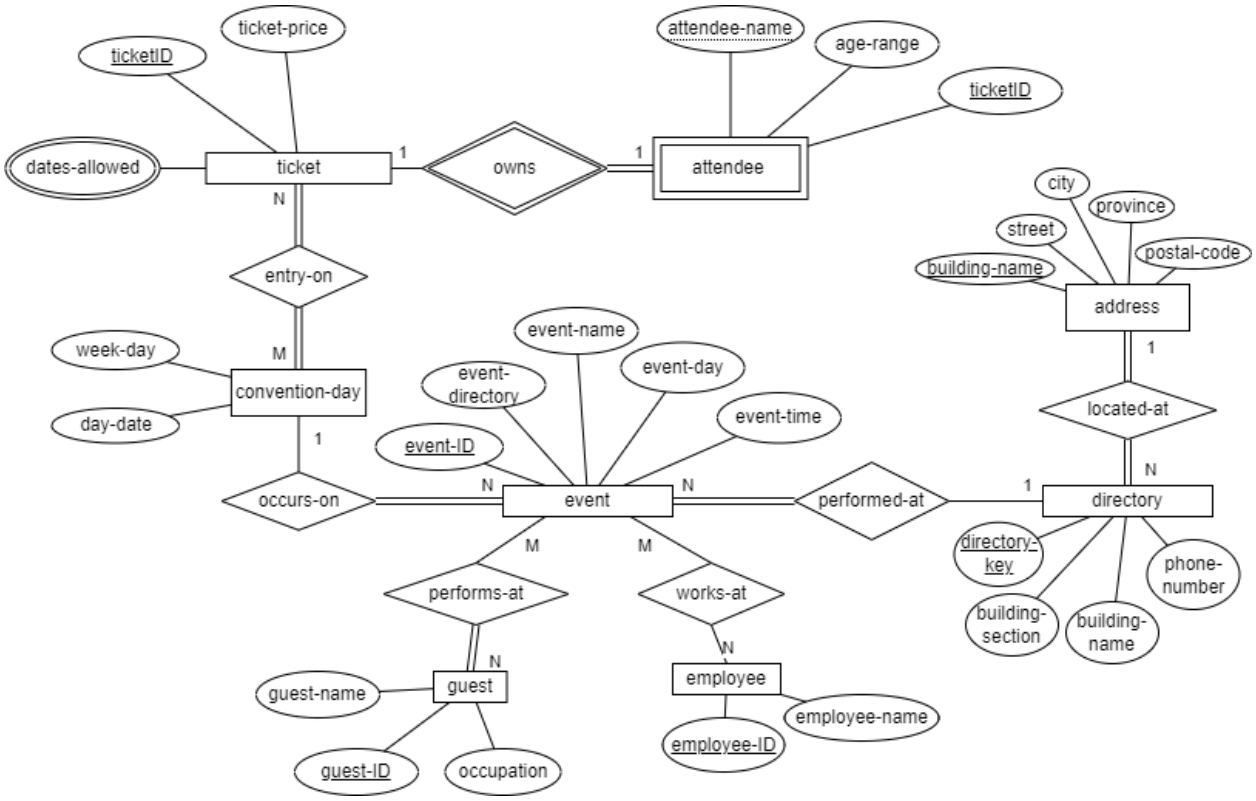
Table of Contents

Introduction

Within the past few years, conventions have grown to become a massive type of event that is present all around the world. Conventions are known to bring a variety of different people together, sharing interests and making memories. From ones about fandoms to conventions about the trades--there's a convention for everyone. While the event alone is a joy to experience, it is important to remember that databases are a key component to ensuring that a convention can proceed smoothly.

Project Documentation

ER Diagram



SQL Tables

Table 1: conventionDay

```
CREATE TABLE conventionDay
    (DayID      NUMBER          PRIMARY KEY,
     DayDate    DATE            NOT NULL,
     WeekDay    VARCHAR(10)     NOT NULL);
```

```
FD = { DayID -> DayDate,
```

```
      DayID -> WeekDay }
```

Table 2: address

```
CREATE TABLE address
    (BuildingName       VARCHAR(30)     PRIMARY KEY,
     Street             VARCHAR(60)     NOT NULL,
     City               VARCHAR(25)     NOT NULL,
     Province           VARCHAR(25)     NOT NULL,
     PostalCode         VARCHAR(7)      NOT NULL);
```

```
FD = { BuildingName -> Street,
      BuildingName -> City,
      BuildingName -> Province
      BuildingName -> PostalCode }
```

Table 3: directory

```
CREATE TABLE directory
    (DirectoryKey       NUMBER          PRIMARY KEY,
     BuildingName       VARCHAR(30)     REFERENCES address(BuildingName) ON DELETE
CASCADE,
     PhoneNumber        VARCHAR(12),
     BuildingSection    VARCHAR(30)     NOT NULL);
```

```
FD = { DirectoryKey -> BuildingName,
      DirectoryKey -> PhoneNumber,
      DirectoryKey -> Building Section }
```

Table 3: event

```
CREATE TABLE event
    (EventID            NUMBER          PRIMARY KEY,
     EventName          VARCHAR(50)     NOT NULL,
     EventDay           NUMBER          NOT NULL    REFERENCES
conventionDay(DayID),
     EventDirectory     NUMBER          NOT NULL    REFERENCES
directory(DirectoryKey),
     EventTime          TIMESTAMP       NOT NULL);
```

```
FD = { EventID -> EventName,
      EventID -> EventDay,
      EventID -> EventDirectory
      EventID -> EventTime }
```

Table 4: guest

```
CREATE TABLE guest
    (GuestID        NUMBER          PRIMARY KEY,
     GuestName      VARCHAR(25)     NOT NULL,
     Occupation     VARCHAR(15));
```

```
FD = { GuestID -> GuestName,
       GuestID -> Occupation }
```

Table 5: employee

```
CREATE TABLE employee
    (EmployeeID     NUMBER          PRIMARY KEY,
     EmployeeName   VARCHAR(25)     NOT NULL);
```

```
FD = { EmployeeID -> EmployeeName }
```

Table 6: performsAt

```
CREATE TABLE performsAt
    (GuestID    NUMBER      REFERENCES guest(GuestID)   ON DELETE CASCADE,
     EventID    NUMBER      REFERENCES event(EventID)   ON DELETE CASCADE,
     PRIMARY KEY (GuestID, EventID));
```

```
FD = {}
```

Table 7: worksAt

```
CREATE TABLE worksAt
    (EmployeeID     NUMBER      REFERENCES employee(EmployeeID) ON DELETE CASCADE,
     EventID        NUMBER      REFERENCES event(EventID)       ON DELETE CASCADE,
     PRIMARY KEY (EmployeeID, EventID));
```

```
FD = {}
```

Table 8: ticket

```
CREATE TABLE ticket
    (TicketID       NUMBER          PRIMARY KEY,
     TicketPrice    DECIMAL(10,2)   NOT NULL);
```

```
FD = { TicketID -> TicketPrice }
```

Table 9: datesAllowed

```
CREATE TABLE datesAllowed
    (TicketID       NUMBER          REFERENCES ticket(TicketID) ON DELETE CASCADE,
     DateAllowed    VARCHAR(10)     NOT NULL,
     PRIMARY KEY (TicketID, DateAllowed));
```

FD = {}

Table 10: attendee

```
CREATE TABLE attendee
    (AttendeeName   VARCHAR(25)     NOT NULL,
     AgeRange       VARCHAR(10)     NOT NULL,
     TicketID       NUMBER          REFERENCES ticket(ticketID) ON DELETE CASCADE,
     PRIMARY KEY (TicketID, AttendeeName));
```

FD = { {TicketID, AttendeeName} -> AgeRange }

Table 11: entryOn

```
CREATE TABLE entryOn
    (TicketID       NUMBER      REFERENCES ticket(ticketID)     ON DELETE CASCADE,
     DayID          NUMBER      REFERENCES conventionDay(DayID) ON DELETE CASCADE,
     PRIMARY KEY (TicketID, DayID));
```

FD = {}

SQL Queries

Query 1: Shows how many attendees are there in each age group, ordered by the number of attendees.

```
SELECT AgeRange AS "Age Group", COUNT(TicketID) AS "Number of People"
FROM ATTENDEE
GROUP BY AgeRange
ORDER BY COUNT(TicketID) DESC;
```

$t_{(\text{AgeRange}\,F_{\text{COUNT TicketID desc}})}\,(P_{\text{“Age Group”, “Number of People”}} \leftarrow \pi_{\text{AgeRange, AgeRange}\,F_{\text{COUNT TicketID}}})(\text{Attendee})$

Query 2: Shows the first and final days of the convention, in addition to the total number of convention days.

```
SELECT MIN(DayDate) AS "First Day", MAX(DayDate) AS "Final Day", COUNT(DISTINCT
DayDate) AS "Convention Length"
FROM CONVENTIONDAY;
```

$P_{\text{“First Day”, “Last Day”, “Convention Length”}} \leftarrow \pi\ F_{\text{MIN DayDate}}, F_{\text{MAX DayDate}}, F_{\text{COUNT }(F_{\text{DISTINCT DayDate}})}(\text{ConventionDay})$

Query 3: Shows a list of ticket IDs that are valid for more than one day, and how many days they’re valid for.

```
SELECT TicketID AS "Multiple Day Ticket IDs", COUNT(DateAllowed) AS "Number of
Days"
FROM DATESALLOWED
GROUP BY TicketID
HAVING COUNT(DateAllowed) > 1;
```

$P_{\text{“Multiple Day Ticket IDs”, “Number of Days”}} \leftarrow \pi\ _{\text{TicketID, TicketID}}F_{\text{COUNT DateAllowed}}(\sigma_{(F_{\text{COUNT DateAllowed > 1}})})(\text{DatesAllowed})$

Query 4: Shows a list of all the unique buildings that the convention has events in.

```
SELECT DISTINCT BuildingName AS "Event Buildings"
FROM DIRECTORY;
```

$P_{\text{“Event Buildings”}} \leftarrow \pi\ F_{\text{DISTINCT BuildingName}}(\text{Directory})$

Query 5: Counts how many employees the convention currently has.

```
SELECT COUNT(EmployeeID) AS "Number of Employees"
FROM EMPLOYEE;
```

$P_{\text{“Number of Employees”}} \leftarrow \pi\ F_{\text{COUNT EmployeeID}}(\text{Employee})$

Query 6: Counts how many tickets provide access to the convention on each day and sorts them by the number of tickets in descending order.

```
SELECT DayID AS "Convention Day ID", COUNT(TicketID) AS "Number of Tickets"
FROM ENTRYON
GROUP BY DayID
ORDER BY COUNT(TicketID) DESC, DayID;
```

$t_{(}F_{\text{COUNT TicketID desc)}}$ $(P_{\text{“Convention Day ID”, “Number of Tickets”}} \text{ <- } \pi_{\text{ DayID, }} F_{\text{COUNT TicketID}})$ (ConventionDay)

Query 7: Counts how many events are being held each day in each directory.

```
SELECT EventDay AS "Convention Day", EventDirectory AS "Location ID",
COUNT(EventID) AS "Number of Events"
FROM EVENT
GROUP BY EventDay, EventDirectory
ORDER BY COUNT(EventID) DESC, EventDay, EventDirectory;
```

$t_{(}F_{\text{COUNT EventID desc)}}$ $(P_{\text{“Convention Day”, “Location ID”, “Number of Events”}} \text{ <- } \pi_{\text{ EventDay, EventDirectory, }} F_{\text{COUNT EventID}})$ (Event)

Query 8: Shows a list of all the unique occupations among the convention’s guests.

```
SELECT DISTINCT Occupation AS "Occupations"
FROM GUEST;
```

$P_{\text{“Occupations”}} \text{ <- } \pi\ F_{\text{DISTINCT Occupation}}$ (Guest)

Query 9: Counts how many events each guest is performing at, listed in descending order.

```
SELECT GuestID AS "Guest IDs", COUNT(EventID) AS "Number of Events Performing At"
FROM PERFORMSAT
GROUP BY GuestID
ORDER BY COUNT(EventID) DESC;
```

$t_{(}F_{\text{COUNT EventID desc)}}$ $(P_{\text{“Guest IDs”, “Number of Events Performing At”}} \text{ <- } \pi_{\text{ GuestID, GuestID}} F_{\text{COUNT EventID}})$ (PerformsAt)

Query 10: Shows how many tickets at each price have been sold, ordered from the most expensive to the least expensive.

```
SELECT TicketPrice AS "Price Point", COUNT(TicketID) AS "Number Sold"
FROM TICKET
GROUP BY TicketPrice
ORDER BY TicketPrice DESC;
```

$t_{\text{TicketPrice desc}}$ $(P_{\text{“Price Point”, “Number Sold”}} \text{ <- } \pi_{\text{ TicketPrice, }} F_{\text{COUNT TicketID}})$ (Ticket)

Query 11: Shows how many employees are working at each event.

```
SELECT EventID AS "Event IDs", COUNT(EmployeeID) AS "Number of Employees"
FROM WORKSAT
GROUP BY EventID
ORDER BY COUNT(EmployeeID) DESC;
```

$t_{(}F_{COUNT\ EmployeeID\ desc)}\ (P_{“Event\ IDs”,\ “Number\ of\ Employees”}\ <-\ \pi_{\ EventID,\ EventID}F_{COUNT\ EmployeeID})\ (WorksAt)$

Query 12: Lists all of the attendees and how many days they will be attending, ordered in a manner of which the people who are attending the most days will appear first.

```
SELECT ATTENDEE.AttendeeName AS "Attendee", COUNT(DATESALLOWED.DateAllowed) AS "Num
Days Attending"
FROM ATTENDEE, TICKET, DATESALLOWED
WHERE ATTENDEE.TicketID = TICKET.TicketID
    AND TICKET.TicketID = DATESALLOWED.TicketID
GROUP BY ATTENDEE.AttendeeName
ORDER BY COUNT(DATESALLOWED.DateAllowed) DESC;
```

$t_{(}F_{COUNT\ DatesAllowed.DateAllowed)}\ (P_{“Attendee”,\ “Num\ Days\ Attending”}\ <-\ \pi_{\ Attendee.AttendeeName,}\ F_{COUNT\ DatesAllowed.DateAllowed}\ (\sigma_{\Theta\ Attendee.TicketID\ =\ Ticket.TicketID\ \wedge\ Ticket.TicketID\ =\ DatesAllowed.TicketID}))(Attendee\ ><_{\Theta}\ Ticket\ ><_{\Theta}\ DatesAllowed)$

Query 13: Lists all of the guests that will be performing on the weekend, and which days of said weekend they're performing on, ordered by date and then date.

```
SELECT DISTINCT GUEST.GuestName AS "Guest", CONVENTIONDAY.WeekDay AS "Performs On"
FROM GUEST, PERFORMSAT, EVENT, CONVENTIONDAY
WHERE GUEST.GuestID = PERFORMSAT.GuestID
    AND PERFORMSAT.EventID = EVENT.EventID
    AND EVENT.EventDay = CONVENTIONDAY.DayID
    AND (CONVENTIONDAY.DayID = 2 OR CONVENTIONDAY.DayID = 3)
ORDER BY CONVENTIONDAY.WeekDay, GUEST.GuestName;
```

$P_{“Guest”,\ “Performs\ On”}\ <-\ \pi\ F_{DISTINCT\ Guest.GuestName,\ ConventionDay.WeekDay}\ (\sigma_{\Theta\ Guest.GuestID\ =\ PerformsAt.GuestID\ \wedge\ PerformsAt.EventID\ =\ ConventionDay.DayID\ \wedge\ (\ \sigma_{ConventionDay.DayID\ =\ 2\ \vee\ ConventionDay.DayID)})(Guest\ ><_{\Theta}\ PerformsAt\ ><_{\Theta}\ Event\ ><_{\Theta}\ ConventionDay)$

Query 14: Lists all of the events happening at the convention, where they're happening, and what day they're happening, ordered by their location and then time.

```
SELECT EVENT.EventName AS "Event", DIRECTORY.BuildingName AS "Building",
DIRECTORY.Section AS "Room", CONVENTIONDAY.WeekDay AS "Day"
FROM EVENT, DIRECTORY, CONVENTIONDAY
WHERE EVENT.EventDirectory = DIRECTORY.DirectoryKey
    AND EVENT.EventDay = CONVENTIONDAY.DayID
ORDER BY DIRECTORY.BuildingName, DIRECTORY.BuildingSection, CONVENTIONDAY.WeekDay;
```

$P_{\text{“Event”, “Building”, “Room”, “Day”}} \leftarrow \pi_{\text{Event.EventName, Directory.BuildingName, Directory.Section, ConventionDay.WeekDay}} (\sigma_{\Theta\ \text{Event.EventDirectory = Directory.DirectoryKey ^ Event.EventDay = ConventionDay.DayID}})(\text{Event} \times\!<_{\Theta} \text{Directory} \times\!<_{\Theta} \text{ConventionDay})$

Query 15: Counts all of the unique events happening at the convention, grouped by the day they’re occurring and ordered by date.

```
SELECT CONVENTIONDAY.WeekDay AS "Day", COUNT(DISTINCT EVENT.EventName) AS "Number
Of Events"
FROM CONVENTIONDAY
INNER JOIN EVENT
ON CONVENTIONDAY.DayID = EVENT.EventDay
GROUP BY CONVENTIONDAY.WeekDay
ORDER BY CONVENTIONDAY.WeekDay;
```

$P_{\text{“Day”, “Number of Events”}} \leftarrow \pi_{\text{ConventionDay.WeekDay, Event.EventDay}} F_{\text{COUNT Event.EventName}} (\sigma_{\Theta\ \text{ConventionDay.DayID = Event.EventDay}})(\text{ConventionDay} \times\!<_{\Theta} \text{Event})$

Query 16: Lists all of the employees working on days 1 and 2 of the convention.

```
SELECT ev.EventDay, e.EmployeeID, e.EmployeeName
FROM employee e, event ev
WHERE EXISTS
(SELECT * FROM worksat w
    WHERE w.EventID = ev.EventID
        AND w.employeeID = e.employeeID
        AND ev.EventDay = 1
)
UNION
(SELECT ev.EventDay, e.EmployeeID, e.EmployeeName
FROM employee e, event ev
WHERE EXISTS
(SELECT * FROM worksat w
    WHERE w.EventID = ev.EventID
        AND w.employeeID = e.employeeID
        AND ev.EventDay = 2
)
```

```
);
```

$(\pi_{\text{Event.EventDay, Employee.EmployeeID, Employee.EmployeeName}}(\text{Employee} >< \text{Event}) \cap (\sigma_{\Theta}$ WorksAt.EventID = Event.EventID ^ WorksAt.EmployeeID = Employee.EmployeeID ^ Event.EventDay = 1) (WorksAt $><_{\Theta}$ Event)) $\cup$ ($\pi_{\text{Event.EventDay, Employee.EmployeeID, Employee.EmployeeName}}$(Employee >< Event) $\cap$ (σ_{Θ} WorksAt.EventID = Event.EventID ^ WorksAt.EmployeeID = Employee.EmployeeID ^ Event.EventDay = 2) (WorksAt $><_{\Theta}$ Event))

Query 17: Displays the average ticket price of each convention day.

```
SELECT CONVENTIONDAY.DayDate AS "Day", AVG(TICKET.TicketPrice) AS "Average Ticket
Price"
FROM CONVENTIONDAY
INNER JOIN ENTRYON
ON CONVENTIONDAY.DayID = ENTRYON.DayID
INNER JOIN TICKET
ON ENTRYON.TicketID = TICKET.TicketID
GROUP BY CONVENTIONDAY.DayDate
ORDER BY CONVENTIONDAY.DayDate;
```

$P_{\text{“Day”, “Average Ticket Price”}}$ <- $\pi_{\text{ConventionDay.DayDate}}$, $F_{\text{AVERAGE Ticket.TicketPrice}}$ (σ_{Θ} ConventionDay.DayID = EntryOn.DayID ^ EntryOn.TicketID = Ticket.TicketID) (ConventionDay $><_{\Theta}$ EntryOn $><_{\Theta}$ Ticket)

Query 18: Lists all of the events that have more than one employee, and how many employees they have.

```
SELECT EVENT.EventName AS "Event", COUNT(EMPLOYEE.EmployeeID) AS "Number of
Employees"
FROM EVENT
INNER JOIN WORKSAT
ON EVENT.EventID = WORKSAT.EventID
INNER JOIN EMPLOYEE
ON WORKSAT.EmployeeID = EMPLOYEE.EmployeeID
GROUP BY Event.EventName
HAVING COUNT(EMPLOYEE.EmployeeID) > 1
ORDER BY COUNT(EMPLOYEE.EmployeeID) DESC;
```

$t_{(F_{\text{COUNT Employee.EmployeeID}})}$ ($P_{\text{“Event”, “Number of Employees”}}$ <- $\pi_{\text{Event.EventName}}$, $F_{\text{COUNT Employee.EmployeeID}}$ (σ_{Θ} Event.EventID = WorksAt.EventID ^ WorksAt.EmployeeID = Employee.EmployeeID ^ $F_{\text{COUNT Employee.EmployeeID > 1}}$)) (Event $><_{\Theta}$ WorksAt $><_{\Theta}$ Employee)

Query 19: Lists all the tickets that are only valid for one day.

```
SELECT e1.TicketID AS "One Day Tickets"
FROM ENTRYON e1
WHERE NOT EXISTS
(SELECT *
FROM ENTRYON e2
WHERE e1.TicketID = e2.TicketID
AND e1.DayID <> e2.DayID);
```

$(P_{\text{"One Day Tickets"}} \leftarrow \pi_{\text{EntryOn1.TicketID}}) \not\emptyset (\sigma_{\Theta\ \text{EntryOn1.TicketID = EntryOn2.TicketID } \wedge \text{ EntryOn1.DayID <> EntryOn2.DayID}}) (\text{EntryOn}_1 >\!\!<_{\Theta} \text{EntryOn}_2)$

Query 20: List Events that don't have an entertainer performing in them.

```
(SELECT e.EventDay, e.EventName
FROM Event e)
MINUS
(SELECT e.EventDay, e.EventName FROM event e, performsat p, guest g
WHERE p.EventID = e.EventID
    AND p.GuestID = g.GuestID
    AND g.Occupation = 'Entertainer'
);
```

$(\pi_{\text{Event.EventDay, Event.EventName}}) \not\emptyset (\pi_{\text{Event.EventDay, Event.EventName}} (\sigma_{\Theta\ \text{PerformsAt.EventID = Event.EventID } \wedge \text{ PerformsAt = Guest.GuestID } \wedge \text{ Guest.Occupation = 'Entertainer'}}) (\text{Event} >\!\!<_{\Theta} \text{PerformsAt} >\!\!<_{\Theta} \text{Guest})$

SQL Views

View 1: Displays all day 2 events, alongside their times and locations.

```
CREATE VIEW EVENT_VIEW AS
SELECT EventName, EventTime, EventDirectory
FROM event
WHERE EventDay = 2;
```

View 2: Displays all convention guests.

```
CREATE VIEW GUEST_VIEW AS
SELECT GuestName, Occupation
FROM guest;
```

View 3: Displays all attendees aged 13 and over.

```
CREATE VIEW ATTENDEE_VIEW AS
SELECT AttendeeName, AgeRange
FROM attendee
WHERE AgeRange = '13+';
```

UNIX Shell Implementation

menu.sh

```
#!/bin/sh

Pause(){
        echo  "Press any key to continue..."
        read WAIT
}

MainMenu()
{

        while [ "$CHOICE" != "START" ]
        do
                clear
                echo
"================================================================="
                echo "|                    Oracle All Inclusive Tool
|"
                echo "|               Main Menu Select Desired Operation(s):
|"
                echo "|         <CTRLZ 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_SELECTEDX X)  Force/Stop/Kill Oracle DB"
                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()
{
        StartMessage
        while [ 1 ]
        do
                MainMenu

        done
}

ProgramStart
```

create_tables.sh

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

CREATE TABLE conventionDay
    (DayID      NUMBER          PRIMARY KEY,
     DayDate    DATE            NOT NULL,
     WeekDay    VARCHAR(10)     NOT NULL);

CREATE TABLE street_city
    (City               VARCHAR(25)     PRIMARY KEY,
     Province           VARCHAR(25)     NOT NULL);

CREATE TABLE postal_street
    (Street             VARCHAR(60)     PRIMARY KEY,
     City               VARCHAR(25)     REFERENCES street_city(City) ON DELETE
CASCADE);

CREATE TABLE building_postal
    (PostalCode         VARCHAR(7)      PRIMARY KEY,
     Street             VARCHAR(60)     REFERENCES postal_street(Street) ON DELETE
CASCADE);

CREATE TABLE directory_building
    (BuildingName       VARCHAR(30)     PRIMARY KEY,
     PostalCode         VARCHAR(7)      REFERENCES building_postal(PostalCode) ON
DELETE CASCADE);

CREATE TABLE directory
    (DirectoryKey       NUMBER          PRIMARY KEY,
     BuildingName       VARCHAR(30)     REFERENCES directory_building(BuildingName)
ON DELETE CASCADE,
     PhoneNumber        VARCHAR(12),
     BuildingSection    VARCHAR(30)     NOT NULL);

CREATE TABLE event
    (EventID            NUMBER          PRIMARY KEY,
     EventName          VARCHAR(50)     NOT NULL,
     EventDay           NUMBER          NOT NULL    REFERENCES
conventionDay(DayID),
     EventDirectory     NUMBER          NOT NULL    REFERENCES
directory(DirectoryKey),
     EventTime          TIMESTAMP       NOT NULL);
```

```
CREATE TABLE guest
    (GuestID        NUMBER          PRIMARY KEY,
     GuestName      VARCHAR(25)     NOT NULL,
     Occupation     VARCHAR(15));

CREATE TABLE employee
    (EmployeeID     NUMBER          PRIMARY KEY,
     EmployeeName   VARCHAR(25)     NOT NULL);

CREATE TABLE performsAt
    (GuestID    NUMBER      REFERENCES guest(GuestID)   ON DELETE CASCADE,
     EventID    NUMBER      REFERENCES event(EventID)   ON DELETE CASCADE,
     PRIMARY KEY (GuestID, EventID));

CREATE TABLE worksAt
    (EmployeeID     NUMBER      REFERENCES employee(EmployeeID) ON DELETE CASCADE,
     EventID        NUMBER      REFERENCES event(EventID)       ON DELETE CASCADE,
     PRIMARY KEY (EmployeeID, EventID));

CREATE TABLE ticket
    (TicketID       NUMBER          PRIMARY KEY,
     TicketPrice    DECIMAL(10,2)   NOT NULL);

CREATE TABLE datesAllowed
    (TicketID       NUMBER          REFERENCES ticket(TicketID) ON DELETE CASCADE,
     DateAllowed    VARCHAR(10)     NOT NULL,
     PRIMARY KEY (TicketID, DateAllowed));

CREATE TABLE attendee
    (AttendeeName   VARCHAR(25)     NOT NULL,
     AgeRange       VARCHAR(10)     NOT NULL,
     TicketID       NUMBER          REFERENCES ticket(ticketID) ON DELETE CASCADE,
     PRIMARY KEY (TicketID, AttendeeName));

CREATE TABLE entryOn
    (TicketID       NUMBER      REFERENCES ticket(ticketID)     ON DELETE CASCADE,
     DayID          NUMBER      REFERENCES conventionDay(DayID) ON DELETE CASCADE,
     PRIMARY KEY (TicketID, DayID));
```

---

drop_tables.sh

```
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64
"USER/PASS@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=15
21))(CONNECT_DATA =(SID=orcl)))" <<EOF
```

```
DROP TABLE entryOn;
DROP TABLE attendee;
DROP TABLE worksAt;
DROP TABLE datesAllowed;
DROP TABLE ticket;
DROP TABLE performsAt;
DROP TABLE employee;
DROP TABLE guest;
DROP TABLE event;
DROP TABLE directory;
DROP TABLE directory_building;
DROP TABLE building_postal;
DROP TABLE postal_street;
DROP TABLE street_city;
DROP TABLE conventionDay;
```

---

populate_tables.sh

```
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64
"USER/PASS@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=15
21))(CONNECT_DATA =(SID=orcl)))" <<EOF

/* Convention Day Data */
INSERT INTO conventionDay (DayID, DayDate, WeekDay)
VALUES (1, TO_DATE('2021-11-05', 'YYYY-MM-DD'), 'Friday');

INSERT INTO conventionDay (DayID, DayDate, WeekDay)
VALUES (2, TO_DATE('2021-11-06', 'YYYY-MM-DD'), 'Saturday');

INSERT INTO conventionDay (DayID, DayDate, WeekDay)
VALUES (3, TO_DATE('2021-11-07', 'YYYY-MM-DD'), 'Sunday');

/* Street-City Data */
INSERT INTO street_city(City, Province)
VALUES ('Toronto', 'Ontario');

/* Postal-Street Data */
INSERT INTO postal_street(Street, City)
VALUES ('Bremner Blvd.', 'Toronto');

INSERT INTO postal_street(Street, City)
VALUES ('Front St.', 'Toronto');
```

```
/* Building-Postal Data */
INSERT INTO building_postal(PostalCode, Street)
VALUES ('M5V 3L9', 'Bremner Blvd.');

INSERT INTO building_postal(PostalCode, Street)
VALUES ('M5V 2W6', 'Front St.');

/* Directory-Building Data */
INSERT INTO directory_building( BuildingName, PostalCode )
VALUES ('MTCC - South Building', 'M5V 3L9');

INSERT INTO directory_building( BuildingName, PostalCode )
VALUES ('MTCC - North Building', 'M5V 2W6');

/* Directory Data */
INSERT INTO directory (DirectoryKey, BuildingName, PhoneNumber, BuildingSection)
VALUES (802, 'MTCC - South Building', '416-585-8000', 'Conference Room');

INSERT INTO directory (DirectoryKey, BuildingName, PhoneNumber, BuildingSection)
VALUES (803, 'MTCC - South Building', '416-585-8000', 'Exhibit Hall DEFG808');

INSERT INTO directory (DirectoryKey, BuildingName, PhoneNumber, BuildingSection)
VALUES (804, 'MTCC - North Building', '416-585-8000', 'Exhibit Hall ABC');

INSERT INTO directory (DirectoryKey, BuildingName, PhoneNumber, BuildingSection)
VALUES (805, 'MTCC - North Building', '416-585-8000', 'John Bassett Theatre');

INSERT INTO directory (DirectoryKey, BuildingName, PhoneNumber, BuildingSection)
VALUES (806, 'MTCC - North Building', '416-585-8000', 'Constitution Hall');

/* Event Data */
INSERT INTO event (EventID, EventName, EventDay, EventDirectory, EventTime)
VALUES (5012, 'Ito Masahiro Autograph Session', 1, 803, TO_TIMESTAMP('2021-11-05
14:30:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO event (EventID, EventName, EventDay, EventDirectory, EventTime)
VALUES (5013, 'Hololive EN 2021 VTuber Tour', 2, 802, TO_TIMESTAMP('2021-11-06
13:00:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO event (EventID, EventName, EventDay, EventDirectory, EventTime)
VALUES (5014, 'The History of TCGs Panel', 2, 806, TO_TIMESTAMP('2021-11-06
16:30:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO event (EventID, EventName, EventDay, EventDirectory, EventTime)
VALUES (5015, 'D4DJ Groovy Mix Production QA', 3, 805, TO_TIMESTAMP('2021-11-06
10:03:00', 'YYYY-MM-DD HH24:MI:SS'));
```

```
/* Guest Data */
INSERT INTO guest (GuestID, GuestName, Occupation)
VALUES (4382947, 'Amelia Watson', 'Entertainer');

INSERT INTO guest (GuestID, GuestName, Occupation)
VALUES (4382948, 'Gawr Gura', 'Entertainer');

INSERT INTO guest (GuestID, GuestName, Occupation)
VALUES (4382949, 'Ninomae Inanis', 'Entertainer');

INSERT INTO guest (GuestID, GuestName, Occupation)
VALUES (4382950, 'Ito Masahiro', 'Voice Actor');

INSERT INTO guest (GuestID, GuestName, Occupation)
VALUES (4382951, 'John Doe', 'Entertainer');

INSERT INTO guest (GuestID, GuestName, Occupation)
VALUES (4382952, 'Jane Doe', 'Game Director');

/* Employee Data */
INSERT INTO employee (EmployeeID, EmployeeName)
VALUES (58009, 'Yoo Jinho');

INSERT INTO employee (EmployeeID, EmployeeName)
VALUES (58010, 'Cinder Ella');

INSERT INTO employee (EmployeeID, EmployeeName)
VALUES (58011, 'Arthur Lounsbery');

INSERT INTO employee (EmployeeID, EmployeeName)
VALUES (58012, 'Loid Forger');

/* Performance Data */
INSERT INTO performsAt (GuestID, EventID)
VALUES (4382950, 5012);

INSERT INTO performsAt (GuestID, EventID)
VALUES (4382947, 5013);

INSERT INTO performsAt (GuestID, EventID)
VALUES (4382947, 5014);

INSERT INTO performsAt (GuestID, EventID)
VALUES (4382947, 5015);

INSERT INTO performsAt (GuestID, EventID)
VALUES (4382948, 5013);
```

```
INSERT INTO performsAt (GuestID, EventID)
VALUES (4382948, 5012);

INSERT INTO performsAt (GuestID, EventID)
VALUES (4382949, 5013);

INSERT INTO performsAt (GuestID, EventID)
VALUES (4382951, 5014);

INSERT INTO performsAt (GuestID, EventID)
VALUES (4382951, 5012);

INSERT INTO performsAt (GuestID, EventID)
VALUES (4382952, 5015);

/* Shift Data */
INSERT INTO worksAt (EmployeeID, EventID)
VALUES (58009, 5012);

INSERT INTO worksAt (EmployeeID, EventID)
VALUES (58009, 5013);

INSERT INTO worksAt (EmployeeID, EventID)
VALUES (58009, 5014);

INSERT INTO worksAt (EmployeeID, EventID)
VALUES (58010, 5013);

INSERT INTO worksAt (EmployeeID, EventID)
VALUES (58011, 5014);

INSERT INTO worksAt (EmployeeID, EventID)
VALUES (58012, 5015);

/* Ticket Data */
INSERT INTO ticket (TicketID, TicketPrice)
VALUES (2024, 55.00);

INSERT INTO ticket (TicketID, TicketPrice)
VALUES (2025, 35.00);

INSERT INTO ticket (TicketID, TicketPrice)
VALUES (2026, 35.00);

INSERT INTO ticket (TicketID, TicketPrice)
VALUES (2027, 25.00);
```

```
INSERT INTO ticket (TicketID, TicketPrice)
VALUES (2028, 10.00);

/* Ticket Allowance Data */
INSERT INTO datesAllowed (TicketID, DateAllowed)
VALUES (2024, 'Saturday');

INSERT INTO datesAllowed (TicketID, DateAllowed)
VALUES (2025, 'Friday');

INSERT INTO datesAllowed (TicketID, DateAllowed)
VALUES (2026, 'Friday');

INSERT INTO datesAllowed (TicketID, DateAllowed)
VALUES (2026, 'Saturday');

INSERT INTO datesAllowed (TicketID, DateAllowed)
VALUES (2027, 'Friday');

INSERT INTO datesAllowed (TicketID, DateAllowed)
VALUES (2027, 'Saturday');

INSERT INTO datesAllowed (TicketID, DateAllowed)
VALUES (2027, 'Sunday');

INSERT INTO datesAllowed (TicketID, DateAllowed)
VALUES (2028, 'Sunday');

/* Attendee Data */
INSERT INTO attendee (AttendeeName, AgeRange, TicketID)
VALUES ('Ren Nanahoshi', '13+', 2024);

INSERT INTO attendee (AttendeeName, AgeRange, TicketID)
VALUES ('Emu Otori', '13+', 2025);

INSERT INTO attendee (AttendeeName, AgeRange, TicketID)
VALUES ('Tsukasa Tenma', '13+', 2026);

INSERT INTO attendee (AttendeeName, AgeRange, TicketID)
VALUES ('Sayu Impact', '0-12', 2027);

INSERT INTO attendee (AttendeeName, AgeRange, TicketID)
VALUES ('Beel Belphegor', '0-12', 2028);

/* Entrance Data */
INSERT INTO entryOn (TicketID, DayID)
```

```
VALUES (2024, 2);

INSERT INTO entryOn (TicketID, DayID)
VALUES (2025, 1);

INSERT INTO entryOn (TicketID, DayID)
VALUES (2026, 1);

INSERT INTO entryOn (TicketID, DayID)
VALUES (2027, 1);

INSERT INTO entryOn (TicketID, DayID)
VALUES (2027, 2);

INSERT INTO entryOn (TicketID, DayID)
VALUES (2027, 3);

INSERT INTO entryOn (TicketID, DayID)
VALUES (2028, 3);
```

---

queries.sh

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

/* Fetch a list of all attendees and how many days they will be attending,
   and order the list so that the people attending the most days show up first */
SELECT ATTENDEE.AttendeeName AS "Attendee", COUNT(DATESALLOWED.DateAllowed) AS "Num
Days Attending"
FROM ATTENDEE
INNER JOIN TICKET
ON ATTENDEE.TicketID = TICKET.TicketID
INNER JOIN DATESALLOWED
ON TICKET.TicketID = DATESALLOWED.TicketID
GROUP BY ATTENDEE.AttendeeName
ORDER BY COUNT(DATESALLOWED.DateAllowed) DESC;

/* Fetch a list of all guests which will be performing on the weekend and which
   weekend day(s) they're performing on, ordered first by day, then by name. */
SELECT DISTINCT GUEST.GuestName AS "Guest", CONVENTIONDAY.WeekDay AS "Performs On"
FROM GUEST
INNER JOIN PERFORMSAT
ON GUEST.GuestID = PERFORMSAT.GuestID
INNER JOIN EVENT
ON PERFORMSAT.EventID = EVENT.EventID
INNER JOIN CONVENTIONDAY
```

```
ON EVENT.EventDay = CONVENTIONDAY.DayID
WHERE (CONVENTIONDAY.DayID = 2 OR CONVENTIONDAY.DayID = 3)
ORDER BY CONVENTIONDAY.WeekDay, GUEST.GuestName;

/* Fetch a list of all events happening at the convention, where they're happening,
   and what day they're happening on. Sorting them first by where they're
happening,
   then by when. */
SELECT EVENT.EventName AS "Event", DIRECTORY.BuildingName AS "Building",
DIRECTORY.BuildingSection AS "Room", CONVENTIOND$
FROM EVENT
INNER JOIN DIRECTORY
ON EVENT.EventDirectory = DIRECTORY.DirectoryKey
INNER JOIN CONVENTIONDAY
ON EVENT.EventDay = CONVENTIONDAY.DayID
ORDER BY DIRECTORY.BuildingName, DIRECTORY.BuildingSection, CONVENTIONDAY.WeekDay;

/* Fetch a count of all unique events happening at the convention, grouped by
   the day they're happening on and ordered by day. */
SELECT CONVENTIONDAY.WeekDay AS "Day", COUNT(DISTINCT EVENT.EventName) AS "Number
Of Events"
FROM CONVENTIONDAY
INNER JOIN EVENT
ON CONVENTIONDAY.DayID = EVENT.EventDay
GROUP BY CONVENTIONDAY.WeekDay
ORDER BY CONVENTIONDAY.WeekDay;

/*--------------------------------------------------------------------------*/
/*--------------------------------------------------------------------------*/
/*--------------------------------------------------------------------------*/

/* List all employees working day 1 and 2 */
SELECT ev.EventDay, e.EmployeeID, e.EmployeeName
FROM employee e, event ev
WHERE EXISTS
(SELECT * FROM worksat w
    WHERE w.EventID = ev.EventID
        AND w.employeeID = e.employeeID
        AND ev.EventDay = 1
)
UNION
(SELECT ev.EventDay, e.EmployeeID, e.EmployeeName
FROM employee e, event ev
WHERE EXISTS
(SELECT * FROM worksat w
    WHERE w.EventID = ev.EventID
        AND w.employeeID = e.employeeID
        AND ev.EventDay = 2
)
);

/* Display the average ticket price of each convention day. */
SELECT CONVENTIONDAY.DayDate AS "Day", AVG(TICKET.TicketPrice) AS "Average Ticket
Price"
```

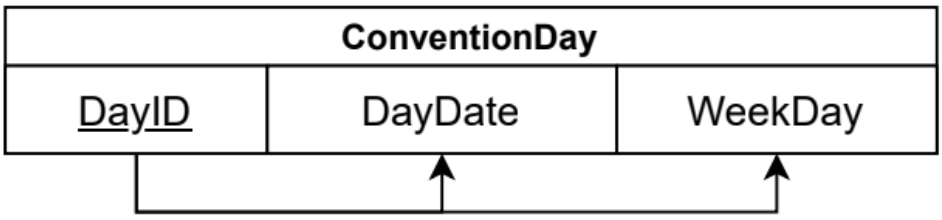
```
FROM CONVENTIONDAY
INNER JOIN ENTRYON
ON CONVENTIONDAY.DayID = ENTRYON.DayID
INNER JOIN TICKET
ON ENTRYON.TicketID = TICKET.TicketID
GROUP BY CONVENTIONDAY.DayDate
ORDER BY CONVENTIONDAY.DayDate;

/* List Events that doesn't have an entertainer performing in them */
(SELECT e.EventDay, e.EventName
FROM Event e)
MINUS
(SELECT e.EventDay, e.EventName FROM event e, performsat p, guest g
WHERE p.EventID = e.EventID
    AND p.GuestID = g.GuestID
    AND g.Occupation = 'Entertainer'
);

/* Fetch a list of all events which have more than one employee, and how many
employees they have */
SELECT EVENT.EventName AS "Event", COUNT(EMPLOYEE.EmployeeID) AS "Number of
Employees"
FROM EVENT
INNER JOIN WORKSAT
ON EVENT.EventID = WORKSAT.EventID
INNER JOIN EMPLOYEE
ON WORKSAT.EmployeeID = EMPLOYEE.EmployeeID
GROUP BY Event.EventName
HAVING COUNT(EMPLOYEE.EmployeeID) > 1
ORDER BY COUNT(EMPLOYEE.EmployeeID) DESC;

/* Query which finds all tickets that are good for only one day. */
SELECT e1.TicketID AS "One Day Tickets"
FROM ENTRYON e1
WHERE NOT EXISTS
(SELECT *
FROM ENTRYON e2
WHERE e1.TicketID = e2.TicketID
AND e1.DayID <> e2.DayID);
```

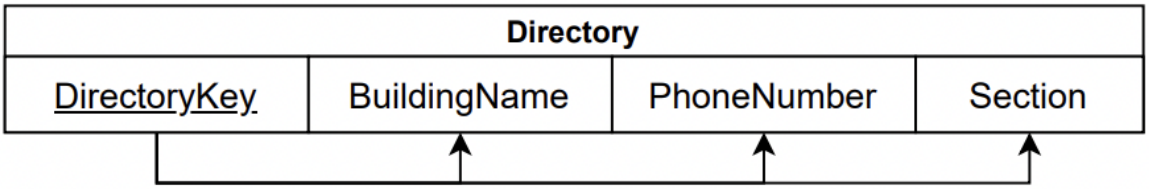
Normalization (3NF, BCNF)



R(DayID, DayDate, WeekDay)
FD = (DayID -> DayDate,

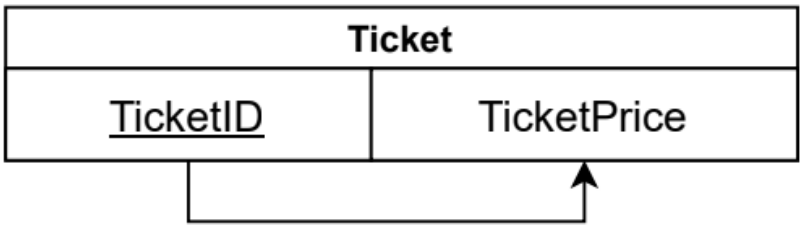
```
      DayID -> WeekDay)
```

This table is in 3NF because all non-key attributes are non-transitively dependent on the primary key.
This table is in BCNF because all FDs have a candidate key to determine dependent attributes.



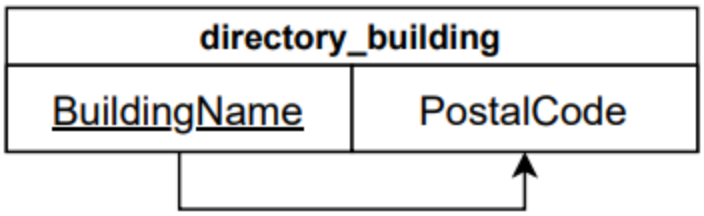

```
R(DirectoryKey, BuildingName, PhoneNumber, Section)
FD = (DirectoryKey -> BuildingName,
      DirectoryKey -> PhoneNumber
      DirectoryKey -> Section)
```

This table is in 3NF because all non-key attributes are non-transitively dependent on the primary key.
This table is in BCNF because all FDs have a candidate key to determine dependent attributes.



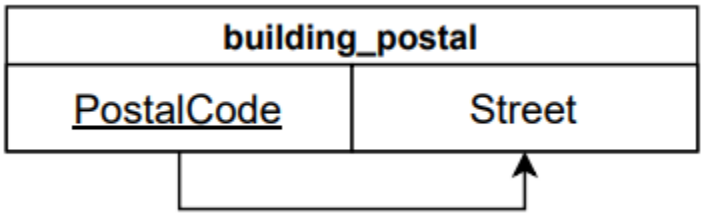

```
R(TicketID, TicketPrice)
FD = (TicketID -> TicketPrice)
```

This table is in 3NF because all non-key attributes are non-transitively dependent on the primary key.
This table is in BCNF because all FDs have a candidate key to determine dependent attributes.



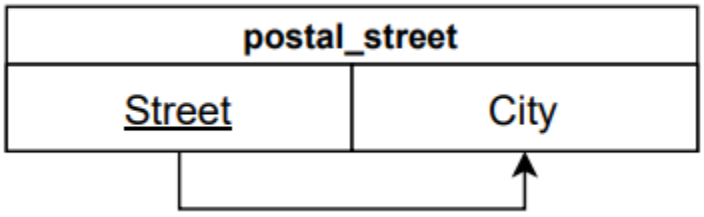
R(BuildingName, PostalCode)
FD = (BuildingName -> PostalCode)

This table is in 3NF because all non-key attributes are non-transitively dependent on the primary key.
This table is in BCNF because all FDs have a candidate key to determine dependent attributes.



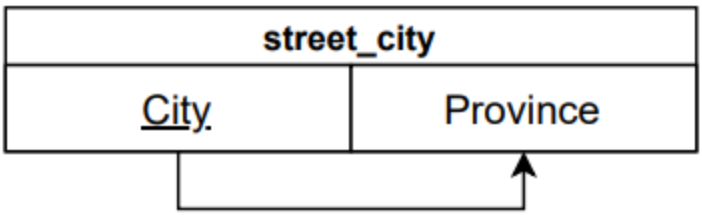
R(PostalCode, Street)
FD = (PostalCode -> Street)

This table is in 3NF because all non-key attributes are non-transitively dependent on the primary key.
This table is in BCNF because all FDs have a candidate key to determine dependent attributes.



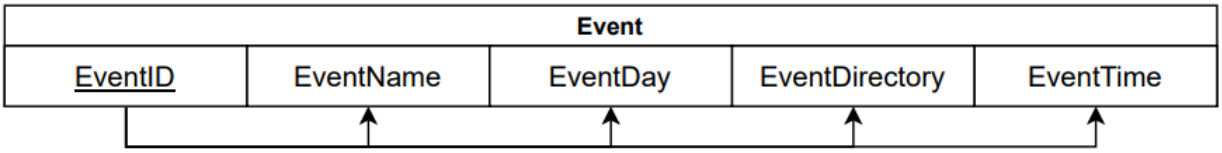
R(Street, City)
FD = (Street -> City)

This table is in 3NF because all non-key attributes are non-transitively dependent on the primary key.
This table is in BCNF because all FDs have a candidate key to determine dependent attributes.



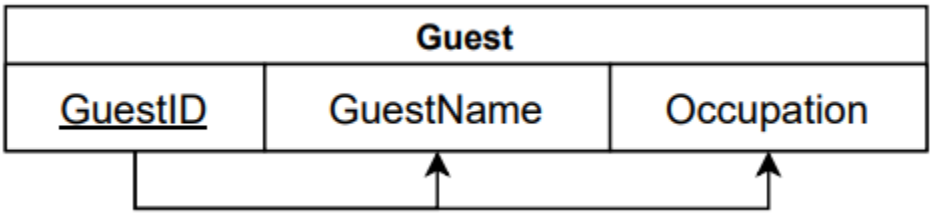
R(City, Province)
FD = (City -> Province)

This table is in 3NF because all non-key attributes are non-transitively dependent on the primary key.
This table is in BCNF because all FDs have a candidate key to determine dependent attributes.



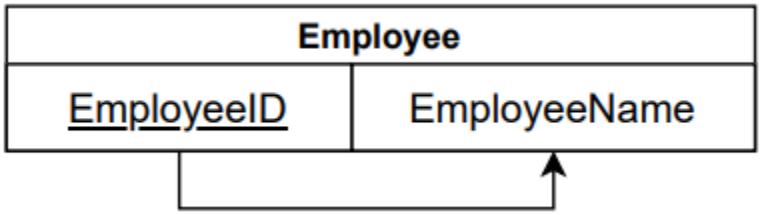
R(EventID, EventName, EventDay, EventDirectory, EventTime)
FD = (EventID -> EventName,
EventID -> EventDay,
EventID -> EventDirectory,
EventID -> EventTime)

This table is in 3NF because all non-key attributes are non-transitively dependent on the primary key.
This table is in BCNF because all FDs have a candidate key to determine dependent attributes.



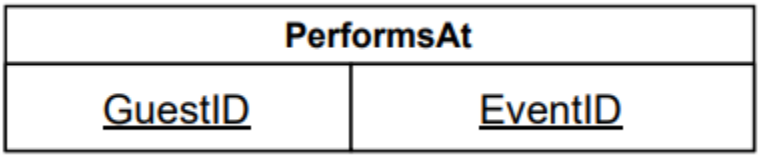
R(GuestID, GuestName, Occupation)
FD = (GuestID -> GuestName,
GuestID -> Occupation)

This table is in 3NF because all non-key attributes are non-transitively dependent on the primary key.
This table is in BCNF because all FDs have a candidate key to determine dependent attributes.



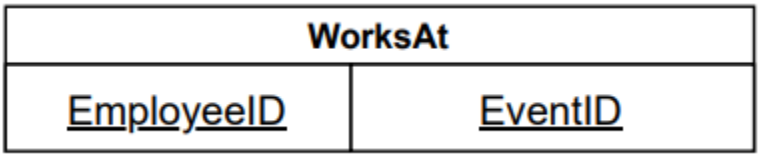

```
R(EmployeeID, EmployeeName)
FD = (EmployeeID -> EmployeeName)
```

This table is in 3NF because all non-key attributes are non-transitively dependent on the primary key.
This table is in BCNF because all FDs have a candidate key to determine dependent attributes.




```
R(GuestID, EventID)
FD = ()
```

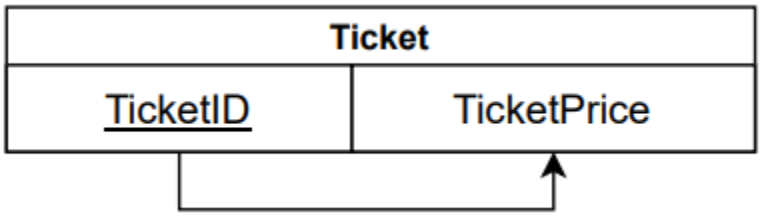
This table is in 3NF because all non-key attributes are non-transitively dependent on the primary key.
This table is in BCNF because all FDs have a candidate key to determine dependent attributes.




```
R(EmployeeID, EventID)
FD = ()
```

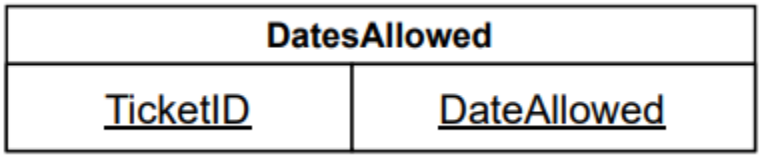
This table is in 3NF because all non-key attributes are non-transitively dependent on the primary key.

This table is in BCNF because all FDs have a candidate key to determine dependent attributes.



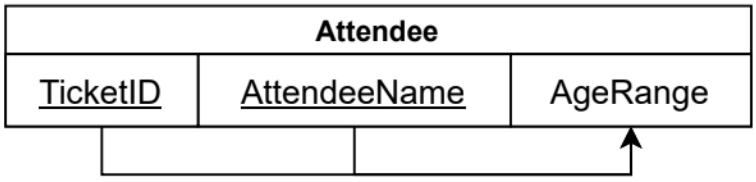
R(TicketID, TicketPrice)
FD = ()

This table is in 3NF because all non-key attributes are non-transitively dependent on the primary key.
This table is in BCNF because all FDs have a candidate key to determine dependent attributes.



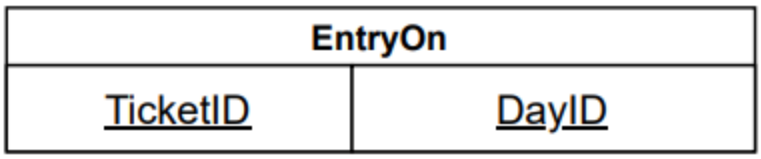
R(TicketID, DateAllowed)
FD = ()

This table is in 3NF because all non-key attributes are non-transitively dependent on the primary key.
This table is in BCNF because all FDs have a candidate key to determine dependent attributes.



R(TicketID, AttendeeName, AgeRange)
FD = (TicketID, AttendeeName -> AgeRange)

This table is in 3NF because all non-key attributes are non-transitively dependent on the primary key.
This table is in BCNF because all FDs have a candidate key to determine dependent attributes.



R(TicketID, DayID)
FD = ()

This table is in 3NF because all non-key attributes are non-transitively dependent on the primary key.
This table is in BCNF because all FDs have a candidate key to determine dependent attributes.

Java Implementation

App.java

```
import java.sql.*;
import java.util.Scanner;

public class App {
       public static Connections connection = new Connections();
       public static CreateTables ctables = new CreateTables();
       public static DeleteTables dtables = new DeleteTables();
       public static PopulateTables ptables = new PopulateTables();
       public static QueryTables qtables = new QueryTables();
       public static boolean exited = false;

    public static void main(String[] args) throws SQLException{
        if (connection.connect()){
            System.out.println("Connected!");

            Scanner in = new Scanner(System.in);;
            String choice;
            while (!exited) {
             System.out.println("M) View Manual\n1) Drop Tables\n2) Create
Tables\n3) Populate Tables\n4) Query Tables\nX) Force/Stop/Kill Oracle DB\nE)
End/Exit\nChoose:");
             choice = in.next();
             switch(choice){

             case "1":
                    dtables.deleteTables(connection);
                    break;

             case "2":
                    ctables.createTables(connection);
                    break;

             case "3":
                    ptables.populateTables(connection);
```

```
                    break;

             case "4":
                    QueryTables.queryTables(connection);
                    break;

             case "X":
                    connection = null;
                    System.out.println("Disconnected from Oracle DB.\n");
                    break;

             case "E":
                    exited = true;
                    break;

             default:
                    break;

             }
            }
            in.close();
         }
    }
}
```

CreateTables.java

```
public class CreateTables {

       public static void createTables(Connections connection) {
              connection.execute("CREATE TABLE conventionDay\n" +
                           "    (DayID      NUMBER          PRIMARY KEY,\n" +
                           "     DayDate    DATE            NOT NULL,\n" +
                           "     WeekDay    VARCHAR(10)     NOT NULL)");
              connection.execute("CREATE TABLE street_city\n" +
                           "    (City               VARCHAR(25)     PRIMARY KEY,\n"
+
                           "     Province           VARCHAR(25)     NOT NULL)");
              connection.execute("CREATE TABLE postal_street\n" +
                           "    (Street             VARCHAR(60)     PRIMARY KEY,\n"
+
                           "     City               VARCHAR(25)     REFERENCES
street_city(City) ON DELETE CASCADE)");
              connection.execute("CREATE TABLE building_postal\n" +
                           "    (PostalCode         VARCHAR(7)      PRIMARY KEY,\n"
+
                           "     Street             VARCHAR(60)     REFERENCES
postal_street(Street) ON DELETE CASCADE)");
              connection.execute("CREATE TABLE directory_building\n" +
                           "    (BuildingName       VARCHAR(30)     PRIMARY KEY,\n"
+
                           "     PostalCode         VARCHAR(7)      REFERENCES
building_postal(PostalCode) ON DELETE CASCADE)");
```

```
            connection.execute("CREATE TABLE directory\n" +
                               "    (DirectoryKey       NUMBER          PRIMARY KEY,\n"
+
                               "     BuildingName       VARCHAR(30)     REFERENCES
directory_building(BuildingName) ON DELETE CASCADE,\n" +
                               "     PhoneNumber        VARCHAR(12),\n" +
                               "     BuildingSection    VARCHAR(30)     NOT NULL)");
            connection.execute("CREATE TABLE event\n" +
                               "    (EventID            NUMBER          PRIMARY KEY,\n"
+
                               "     EventName          VARCHAR(50)     NOT NULL,\n" +
                               "     EventDay           NUMBER          NOT NULL
REFERENCES conventionDay(DayID),\n" +
                               "     EventDirectory     NUMBER          NOT NULL
REFERENCES directory(DirectoryKey),\n" +
                               "     EventTime          TIMESTAMP       NOT NULL)");
            connection.execute("CREATE TABLE guest\n" +
                               "    (GuestID        NUMBER          PRIMARY KEY,\n" +
                               "     GuestName      VARCHAR(25)     NOT NULL,\n" +
                               "     Occupation     VARCHAR(15))");
            connection.execute("CREATE TABLE employee\n" +
                               "    (EmployeeID     NUMBER          PRIMARY KEY,\n" +
                               "     EmployeeName   VARCHAR(25)     NOT NULL)");
            connection.execute("CREATE TABLE performsAt\n" +
                               "    (GuestID    NUMBER      REFERENCES guest(GuestID)
ON DELETE CASCADE,\n" +
                               "     EventID    NUMBER      REFERENCES event(EventID)
ON DELETE CASCADE,\n" +
                               "     PRIMARY KEY (GuestID, EventID))");
            connection.execute("CREATE TABLE worksAt\n" +
                               "    (EmployeeID     NUMBER      REFERENCES
employee(EmployeeID) ON DELETE CASCADE,\n" +
                               "     EventID        NUMBER      REFERENCES
event(EventID)       ON DELETE CASCADE,\n" +
                               "     PRIMARY KEY (EmployeeID, EventID))");
            connection.execute("CREATE TABLE ticket\n" +
                               "    (TicketID       NUMBER          PRIMARY KEY,\n" +
                               "     TicketPrice    DECIMAL(10,2)   NOT NULL)");
            connection.execute("CREATE TABLE datesAllowed\n" +
                               "    (TicketID       NUMBER          REFERENCES
ticket(TicketID) ON DELETE CASCADE,\n" +
                               "     DateAllowed    VARCHAR(10)     NOT NULL,\n" +
                               "     PRIMARY KEY (TicketID, DateAllowed))");
            connection.execute("CREATE TABLE attendee\n" +
                               "    (AttendeeName   VARCHAR(25)     NOT NULL,\n" +
                               "     AgeRange       VARCHAR(10)     NOT NULL,\n" +
                               "     TicketID       NUMBER          REFERENCES
ticket(ticketID) ON DELETE CASCADE,\n" +
                               "     PRIMARY KEY (TicketID, AttendeeName))");
            connection.execute("CREATE TABLE entryOn\n" +
                               "    (TicketID       NUMBER      REFERENCES
ticket(ticketID)     ON DELETE CASCADE,\n" +
                               "     DayID          NUMBER      REFERENCES
conventionDay(DayID) ON DELETE CASCADE,\n" +
```

```
                        "     PRIMARY KEY (TicketID, DayID))");

            System.out.println("All Tables Created!\n");
      }
}
```

DeleteTables.java

```
public class DeleteTables {
      public static void deleteTables(Connections connection) {
            connection.execute("DROP TABLE entryOn");
            connection.execute("DROP TABLE attendee");
            connection.execute("DROP TABLE worksAt");
            connection.execute("DROP TABLE datesAllowed");
            connection.execute("DROP TABLE ticket");
            connection.execute("DROP TABLE performsAt");
            connection.execute("DROP TABLE employee");
            connection.execute("DROP TABLE guest");
            connection.execute("DROP TABLE event");
            connection.execute("DROP TABLE directory");
            connection.execute("DROP TABLE directory_building");
            connection.execute("DROP TABLE building_postal");
            connection.execute("DROP TABLE postal_street");
            connection.execute("DROP TABLE street_city");
            connection.execute("DROP TABLE conventionDay");

            System.out.println("All Tables Deleted!\n");
      }
}
```

PopulateTables.java

```
public class PopulateTables {
      public static void populateTables(Connections connection) {
            connection.execute("INSERT INTO conventionDay (DayID, DayDate,
WeekDay)\n" +
                        "VALUES (1, TO_DATE('2021-11-05', 'YYYY-MM-DD'),
'Friday')");
            connection.execute("INSERT INTO conventionDay (DayID, DayDate,
WeekDay)\n" +
                        "VALUES (2, TO_DATE('2021-11-06', 'YYYY-MM-DD'),
'Saturday')");
            connection.execute("INSERT INTO conventionDay (DayID, DayDate,
WeekDay)\n" +
                        "VALUES (3, TO_DATE('2021-11-07', 'YYYY-MM-DD'),
'Sunday')");

            connection.execute("INSERT INTO street_city(City, Province)\n" +
                        "VALUES ('Toronto', 'Ontario')");

            connection.execute("INSERT INTO postal_street(Street, City)\n" +
                        "VALUES ('Bremner Blvd.', 'Toronto')");
```

```
            connection.execute("INSERT INTO postal_street(Street, City)\n" +
                        "VALUES ('Front St.', 'Toronto')");

            connection.execute("INSERT INTO building_postal(PostalCode, Street)\n"
+
                        "VALUES ('M5V 3L9', 'Bremner Blvd.')");
            connection.execute("INSERT INTO building_postal(PostalCode, Street)\n"
+
                        "VALUES ('M5V 2W6', 'Front St.')");

            connection.execute("INSERT INTO directory_building( BuildingName,
PostalCode )\n" +
                        "VALUES ('MTCC - South Building', 'M5V 3L9')");
            connection.execute("INSERT INTO directory_building( BuildingName,
PostalCode )\n" +
                        "VALUES ('MTCC - North Building', 'M5V 2W6')");

            connection.execute("INSERT INTO directory (DirectoryKey, BuildingName,
PhoneNumber, BuildingSection)\n" +
                        "VALUES (802, 'MTCC - South Building', '416-585-8000',
'Conference Room')");
            connection.execute("INSERT INTO directory (DirectoryKey, BuildingName,
PhoneNumber, BuildingSection)\n" +
                        "VALUES (803, 'MTCC - South Building', '416-585-8000',
'Exhibit Hall DEFG808')");
            connection.execute("INSERT INTO directory (DirectoryKey, BuildingName,
PhoneNumber, BuildingSection)\n" +
                        "VALUES (804, 'MTCC - North Building', '416-585-8000',
'Exhibit Hall ABC')");
            connection.execute("INSERT INTO directory (DirectoryKey, BuildingName,
PhoneNumber, BuildingSection)\n" +
                        "VALUES (805, 'MTCC - North Building', '416-585-8000',
'John Bassett Theatre')");
            connection.execute("INSERT INTO directory (DirectoryKey, BuildingName,
PhoneNumber, BuildingSection)\n" +
                        "VALUES (806, 'MTCC - North Building', '416-585-8000',
'Constitution Hall')");

            connection.execute("INSERT INTO event (EventID, EventName, EventDay,
EventDirectory, EventTime)\n" +
                        "VALUES (5012, 'Ito Masahiro Autograph Session', 1, 803,
TO_TIMESTAMP('2021-11-05 14:30:00', 'YYYY-MM-DD HH24:MI:SS'))");
            connection.execute("INSERT INTO event (EventID, EventName, EventDay,
EventDirectory, EventTime)\n" +
                        "VALUES (5013, 'Hololive EN 2021 VTuber Tour', 2, 802,
TO_TIMESTAMP('2021-11-06 13:00:00', 'YYYY-MM-DD HH24:MI:SS'))");
            connection.execute("INSERT INTO event (EventID, EventName, EventDay,
EventDirectory, EventTime)\n" +
                        "VALUES (5014, 'The History of TCGs Panel', 2, 806,
TO_TIMESTAMP('2021-11-06 16:30:00', 'YYYY-MM-DD HH24:MI:SS'))");
            connection.execute("INSERT INTO event (EventID, EventName, EventDay,
EventDirectory, EventTime)\n" +
                        "VALUES (5015, 'D4DJ Groovy Mix Production QA', 3, 805,
TO_TIMESTAMP('2021-11-06 10:03:00', 'YYYY-MM-DD HH24:MI:SS'))");
```

```
            connection.execute("INSERT INTO guest (GuestID, GuestName,
Occupation)\n" +
                               "VALUES (4382947, 'Amelia Watson', 'Entertainer')");
            connection.execute("INSERT INTO guest (GuestID, GuestName,
Occupation)\n" +
                               "VALUES (4382948, 'Gawr Gura', 'Entertainer')");
            connection.execute("INSERT INTO guest (GuestID, GuestName,
Occupation)\n" +
                               "VALUES (4382949, 'Ninomae Inanis', 'Entertainer')");
            connection.execute("INSERT INTO guest (GuestID, GuestName,
Occupation)\n" +
                               "VALUES (4382950, 'Ito Masahiro', 'Voice Actor')");
            connection.execute("INSERT INTO guest (GuestID, GuestName,
Occupation)\n" +
                               "VALUES (4382951, 'John Doe', 'Entertainer')");
            connection.execute("INSERT INTO guest (GuestID, GuestName,
Occupation)\n" +
                               "VALUES (4382952, 'Jane Doe', 'Game Director')");

            connection.execute("INSERT INTO employee (EmployeeID, EmployeeName)\n"
+
                               "VALUES (58009, 'Yoo Jinho')");
            connection.execute("INSERT INTO employee (EmployeeID, EmployeeName)\n"
+
                               "VALUES (58010, 'Cinder Ella')");
            connection.execute("INSERT INTO employee (EmployeeID, EmployeeName)\n"
+
                               "VALUES (58011, 'Arthur Lounsbery')");
            connection.execute("INSERT INTO employee (EmployeeID, EmployeeName)\n"
+
                               "VALUES (58012, 'Loid Forger')");

            connection.execute("INSERT INTO performsAt (GuestID, EventID)\n" +
                               "VALUES (4382950, 5012)");
            connection.execute("INSERT INTO performsAt (GuestID, EventID)\n" +
                               "VALUES (4382947, 5013)");
            connection.execute("INSERT INTO performsAt (GuestID, EventID)\n" +
                               "VALUES (4382947, 5014)");
            connection.execute("INSERT INTO performsAt (GuestID, EventID)\n" +
                               "VALUES (4382947, 5015)");
            connection.execute("INSERT INTO performsAt (GuestID, EventID)\n" +
                               "VALUES (4382948, 5013)");
            connection.execute("INSERT INTO performsAt (GuestID, EventID)\n" +
                               "VALUES (4382948, 5012)");
            connection.execute("INSERT INTO performsAt (GuestID, EventID)\n" +
                               "VALUES (4382949, 5013)");
            connection.execute("INSERT INTO performsAt (GuestID, EventID)\n" +
                               "VALUES (4382951, 5014)");
            connection.execute("INSERT INTO performsAt (GuestID, EventID)\n" +
                               "VALUES (4382951, 5012)");
            connection.execute("INSERT INTO performsAt (GuestID, EventID)\n" +
                               "VALUES (4382952, 5015)");
```

```
            connection.execute("INSERT INTO worksAt (EmployeeID, EventID)\n" +
                        "VALUES (58009, 5012)");
            connection.execute("INSERT INTO worksAt (EmployeeID, EventID)\n" +
                        "VALUES (58009, 5013)");
            connection.execute("INSERT INTO worksAt (EmployeeID, EventID)\n" +
                        "VALUES (58009, 5014)");
            connection.execute("INSERT INTO worksAt (EmployeeID, EventID)\n" +
                        "VALUES (58010, 5013)");
            connection.execute("INSERT INTO worksAt (EmployeeID, EventID)\n" +
                        "VALUES (58011, 5014)");
            connection.execute("INSERT INTO worksAt (EmployeeID, EventID)\n" +
                        "VALUES (58012, 5015)");

            connection.execute("INSERT INTO ticket (TicketID, TicketPrice)\n" +
                        "VALUES (2024, 55.00)");
            connection.execute("INSERT INTO ticket (TicketID, TicketPrice)\n" +
                        "VALUES (2025, 35.00)");
            connection.execute("INSERT INTO ticket (TicketID, TicketPrice)\n" +
                        "VALUES (2026, 35.00)");
            connection.execute("INSERT INTO ticket (TicketID, TicketPrice)\n" +
                        "VALUES (2027, 25.00)");
            connection.execute("INSERT INTO ticket (TicketID, TicketPrice)\n" +
                        "VALUES (2028, 10.00)");

            connection.execute("INSERT INTO datesAllowed (TicketID,
DateAllowed)\n" +
                        "VALUES (2024, 'Saturday')");
            connection.execute("INSERT INTO datesAllowed (TicketID,
DateAllowed)\n" +
                        "VALUES (2025, 'Friday')");
            connection.execute("INSERT INTO datesAllowed (TicketID,
DateAllowed)\n" +
                        "VALUES (2026, 'Friday')");
            connection.execute("INSERT INTO datesAllowed (TicketID,
DateAllowed)\n" +
                        "VALUES (2026, 'Saturday')");
            connection.execute("INSERT INTO datesAllowed (TicketID,
DateAllowed)\n" +
                        "VALUES (2027, 'Friday')");
            connection.execute("INSERT INTO datesAllowed (TicketID,
DateAllowed)\n" +
                        "VALUES (2027, 'Saturday')");
            connection.execute("INSERT INTO datesAllowed (TicketID,
DateAllowed)\n" +
                        "VALUES (2027, 'Sunday')");
            connection.execute("INSERT INTO datesAllowed (TicketID,
DateAllowed)\n" +
                        "VALUES (2028, 'Sunday')");

            connection.execute("INSERT INTO attendee (AttendeeName, AgeRange,
TicketID)\n" +
                        "VALUES ('Ren Nanahoshi', '13+', 2024)");
            connection.execute("INSERT INTO attendee (AttendeeName, AgeRange,
TicketID)\n" +
```

```
                        "VALUES ('Emu Otori', '13+', 2025)");
            connection.execute("INSERT INTO attendee (AttendeeName, AgeRange,
TicketID)\n" +
                        "VALUES ('Tsukasa Tenma', '13+', 2026)");
            connection.execute("INSERT INTO attendee (AttendeeName, AgeRange,
TicketID)\n" +
                        "VALUES ('Sayu Impact', '0-12', 2027)");
            connection.execute("INSERT INTO attendee (AttendeeName, AgeRange,
TicketID)\n" +
                        "VALUES ('Beel Belphegor', '0-12', 2028)");

            connection.execute("INSERT INTO entryOn (TicketID, DayID)\n" +
                        "VALUES (2024, 2)");
            connection.execute("INSERT INTO entryOn (TicketID, DayID)\n" +
                        "VALUES (2025, 1)");
            connection.execute("INSERT INTO entryOn (TicketID, DayID)\n" +
                        "VALUES (2026, 1)");
            connection.execute("INSERT INTO entryOn (TicketID, DayID)\n" +
                        "VALUES (2027, 1)");
            connection.execute("INSERT INTO entryOn (TicketID, DayID)\n" +
                        "VALUES (2027, 2)");
            connection.execute("INSERT INTO entryOn (TicketID, DayID)\n" +
                        "VALUES (2027, 3)");
            connection.execute("INSERT INTO entryOn (TicketID, DayID)\n" +
                        "VALUES (2028, 3)");

            System.out.println("All Tables Populated!\n");
      }
}
```

QueryTables.java

```
import java.sql.ResultSet;
import java.sql.SQLException;

public class QueryTables {
      public static ResultSet result = null;
      public static void queryTables(Connections connection) throws SQLException {
            result = connection.execute("SELECT AgeRange AS \"Age Group\",
COUNT(TicketID) AS \"Number of People\"\n" +
                        "FROM ATTENDEE\n" +
                        "GROUP BY AgeRange\n" +
                        "ORDER BY COUNT(TicketID) DESC");
            if (result != null) {
                  System.out.println("1. How many people are attending from each
age group");
                  System.out.println("Age Group\tNumber of People");
                  while(result.next()) {
                        System.out.println(result.getString("Age
Group")+"\t\t"+result.getString("Number of People"));
                  }
            }
            System.out.println();
```

```
            result = connection.execute("SELECT MIN(DayDate) AS \"First Day\",
MAX(DayDate) AS \"Final Day\", COUNT(DISTINCT DayDate) AS \"Convention Length\"\n"
+
                        "FROM CONVENTIONDAY");

            if (result != null) {
                  System.out.println("2. Display the first day and the last day
of the convention");
                  System.out.println("First Day\t\t\tFinal Day\t\t\tConvention
Length");
                  while(result.next()) {
                        System.out.println(result.getString("First
Day")+"\t\t"+result.getString("Final Day")+"\t\t"+result.getString("Convention
Length"));
                  }
            }
            System.out.println();

            result = connection.execute("SELECT e1.TicketID AS \"One Day
Tickets\"\n" +
                        "FROM ENTRYON e1\n" +
                        "WHERE NOT EXISTS\n" +
                        "(SELECT *\n" +
                        "FROM ENTRYON e2\n" +
                        "WHERE e1.TicketID = e2.TicketID\n" +
                        "AND e1.DayID <> e2.DayID)");

            if (result != null) {
                  System.out.println("3. Displays all the available one day
tickets");
                  System.out.println("One Day Tickets");
                  while(result.next()) {
                        System.out.println(result.getString("One Day Tickets"));
                  }
            }
            System.out.println();

            result = connection.execute("SELECT COUNT(EmployeeID) AS \"Number of
Employees\"\n" +
                        "FROM EMPLOYEE");
            if (result != null) {
                 System.out.println("4. Total number of employees working in the
convention");
                 System.out.println("Number of Employees");
                 while(result.next()) {
                     System.out.println(result.getString("Number of employees"));
                 }
                 }
                 System.out.println();

            result = connection.execute("SELECT CONVENTIONDAY.DayDate AS \"Day\",
AVG(TICKET.TicketPrice) AS \"Average Ticket Price\"\n" +
                   "FROM CONVENTIONDAY\n" +
```

```
                    "INNER JOIN ENTRYON\n" +
                    "ON CONVENTIONDAY.DayID = ENTRYON.DayID\n" +
                    "INNER JOIN TICKET\n" +
                    "ON ENTRYON.TicketID = TICKET.TicketID\n" +
                    "GROUP BY CONVENTIONDAY.DayDate\n" +
                    "ORDER BY CONVENTIONDAY.DayDate");
            if (result != null) {
                  System.out.println("5. Average ticket price for each day");
                  System.out.println("Day\t\t\tAverage Ticket Trice");
           while(result.next()) {
               System.out.println(result.getString("Day")+ "\t" +
result.getString("Average Ticket Price"));
           }
       }
       System.out.println();

            result = connection.execute("SELECT TicketID AS \"Multiple Day Ticket
IDs\", COUNT(DateAllowed) AS \"Number of Days\"\n" +
                        "FROM DATESALLOWED\n" +
                        "GROUP BY TicketID\n" +
                        "HAVING COUNT(DateAllowed) > 1");
            if (result != null) {
                  System.out.println("6. Displays all tickets that work on
multiple days and the number of days they work.");
                  System.out.println("Multiple Day Ticket IDs\tNumber of Day");
           while(result.next()) {
               System.out.println(result.getString("Multiple Day Ticket IDs")+
"\t\t\t" + result.getString("Number of Days"));
           }
       }
       System.out.println();

            result = connection.execute("SELECT DISTINCT BuildingName AS \"Event
Buildings\"\n" +
                        "FROM DIRECTORY");
            if (result != null) {
                  System.out.println("7. Displays all the buildings for the
conventions");
                  System.out.println("Event Buildings");
           while(result.next()) {
               System.out.println(result.getString("Event Buildings"));
           }
       }
       System.out.println();

       result = connection.execute("SELECT EVENT.EventName AS \"Event\",
COUNT(EMPLOYEE.EmployeeID) AS \"Number of Employees\"\n" +
                  "FROM EVENT\n" +
                  "INNER JOIN WORKSAT\n" +
                  "ON EVENT.EventID = WORKSAT.EventID\n" +
                  "INNER JOIN EMPLOYEE\n" +
                  "ON WORKSAT.EmployeeID = EMPLOYEE.EmployeeID\n" +
                  "GROUP BY Event.EventName\n" +
                  "HAVING COUNT(EMPLOYEE.EmployeeID) > 1\n" +
```

```
                    "ORDER BY COUNT(EMPLOYEE.EmployeeID) DESC");
        if (result != null) {
            System.out.println("8. List of all events which have more than one
employee");
            System.out.println("Event\t\t\t\tNumber of Employees");
            while(result.next()) {

System.out.println(result.getString("Event")+"\t"+result.getString("Number of
Employees"));
            }
        }
        System.out.println();

        result = connection.execute("SELECT ATTENDEE.AttendeeName AS \"Attendee\",
COUNT(DATESALLOWED.DateAllowed) AS \"Num Days Attending\"\n"+
                "FROM ATTENDEE\n"+
                "INNER JOIN TICKET\n"+
                "ON ATTENDEE.TicketID = TICKET.TicketID\n"+
                "INNER JOIN DATESALLOWED\n"+
                "ON TICKET.TicketID = DATESALLOWED.TicketID\n"+
                "GROUP BY ATTENDEE.AttendeeName\n"+
                "ORDER BY COUNT(DATESALLOWED.DateAllowed) DESC\n");

        if (result != null) {
                System.out.println("9. Attendees and the number of days they're
attending, in descending order.");
                System.out.println("Attendee\tNum Days Attending");
                    while(result.next()) {
                        System.out.println(result.getString("Attendee")+ "\t" +
result.getString("Num Days Attending"));
                    }
                }
        System.out.println();

             result = connection.execute("SELECT DayID AS \"Convention Day ID\",
COUNT(TicketID) AS \"Number of Tickets\"\n" +
                            "FROM ENTRYON\n" +
                            "GROUP BY DayID\n" +
                            "ORDER BY COUNT(TicketID) DESC, DayID");

             if (result != null) {
            System.out.println("10. Displays the number of tickets that work for
each convention day.");
            System.out.println("Convention Day ID\tNumber of Tickets");
                while(result.next()) {
                    System.out.println(result.getString("Convention Day ID")+
"\t\t\t" + result.getString("Number of Tickets"));
                }
            }
             System.out.println();

             result = connection.execute("SELECT DISTINCT Occupation AS
\"Occupations\"\n" +
                            "FROM GUEST");
```

```
             if (result != null) {
            System.out.println("11. Displays all the different occupations of
guests.");
            System.out.println("Occupations");
                while(result.next()) {
                    System.out.println(result.getString("Occupations"));
                }
            }
             System.out.println();

             result = connection.execute("SELECT DISTINCT GUEST.GuestName AS
\"Guest\", CONVENTIONDAY.WeekDay AS \"Performs On\""+
                   "FROM GUEST\n"+
                   "INNER JOIN PERFORMSAT\n"+
                   "ON GUEST.GuestID = PERFORMSAT.GuestID\n"+
                   "INNER JOIN EVENT\n"+
                   "ON PERFORMSAT.EventID = EVENT.EventID\n"+
                   "INNER JOIN CONVENTIONDAY\n"+
                   "ON EVENT.EventDay = CONVENTIONDAY.DayID\n"+
                   "WHERE (CONVENTIONDAY.DayID = 2 OR CONVENTIONDAY.DayID = 3)\n"+
                   "ORDER BY CONVENTIONDAY.WeekDay, GUEST.GuestName\n");

              if (result != null) {
                  System.out.println("12. A list of all weekend guests and which
day they'll be performing on, sorted by day, then name.");
                  System.out.println("Guest\t\tPerforms On");
                      while(result.next()) {
                          System.out.println(result.getString("Guest")+ "\t" +
result.getString("Performs On"));
                      }
                  }
              System.out.println();

             result = connection.execute("SELECT GuestID AS \"Guest IDs\",
COUNT(EventID) AS \"Number of Events Performing At\"\n" +
                           "FROM PERFORMSAT\n" +
                           "GROUP BY GuestID\n" +
                           "ORDER BY COUNT(EventID) DESC");
             if (result != null) {
            System.out.println("13. Displays all the guests and the number of
events they are performing in.");
            System.out.println("Guest IDs\tNumber of Events Performing At");
                while(result.next()) {
                    System.out.println(result.getString("Guest IDs")+ "\t\t" +
result.getString("Number of Events Performing At"));
                }
            }
        System.out.println();

        result = connection.execute("SELECT EventDay AS \"Convention Day\",
EventDirectory AS \"Location ID\", COUNT(EventID) AS \"Number of Events\"\n" +
                   "FROM EVENT\n" +
                   "GROUP BY EventDay, EventDirectory\n" +
```

```
                    "ORDER BY COUNT(EventID) DESC, EventDay, EventDirectory");
        if (result != null) {
            System.out.println("14. Count how many events are being held each day
in each directory");
            System.out.println("Convention Day\tLocation ID\tNumber of Events");
            while(result.next()) {
                System.out.println(result.getString("Convention
Day")+"\t\t"+result.getString("Location ID")+"\t\t"+result.getString("Number of
Events"));
            }
        }
        System.out.println();

        result = connection.execute("SELECT EVENT.EventName AS \"Event\",
DIRECTORY.BuildingName AS \"Building\", DIRECTORY.BuildingSection AS \"Room\",
CONVENTIONDAY.WeekDay AS \"Day\"\n"+
                "FROM EVENT\n"+
                "INNER JOIN DIRECTORY\n"+
                "ON EVENT.EventDirectory = DIRECTORY.DirectoryKey\n"+
                "INNER JOIN CONVENTIONDAY\n"+
                "ON EVENT.EventDay = CONVENTIONDAY.DayID\n"+
                "ORDER BY DIRECTORY.BuildingName, DIRECTORY.BuildingSection,
CONVENTIONDAY.WeekDay");

        if (result != null) {
                System.out.println("15. A list of all events happening at the
convention, where they're happening, and when they're happening, sorted by where
they're happening, then by when.");
                System.out.println("Event\t\t\t\tBuilding\t\tRoom\t\t\tDay");
                    while(result.next()) {
                        System.out.println(result.getString("Event")+ "\t" +
result.getString("Building")+ "\t" + result.getString("Room")+ "\t" +
result.getString("Day"));
                    }
                }
        System.out.println();

        result = connection.execute("SELECT TicketPrice AS \"Price Point\",
COUNT(TicketID) AS \"Number Sold\"\n" +
                    "FROM TICKET\n" +
                    "GROUP BY TicketPrice\n" +
                    "ORDER BY TicketPrice DESC");
        if (result != null) {
                System.out.println("16. How many tickets of each price point have
been sold");
                System.out.println("Price Point\tNumber Sold");
                while(result.next()) {
                    System.out.println(result.getString("Price
Point")+"\t\t"+result.getString("Number Sold"));
                }
            }
        System.out.println();

        result = connection.execute("SELECT CONVENTIONDAY.WeekDay AS \"Day\",
```

```
COUNT(DISTINCT EVENT.EventName) AS \"Number Of Events\"\n"+
                "FROM CONVENTIONDAY\n"+
                "INNER JOIN EVENT\n"+
                "ON CONVENTIONDAY.DayID = EVENT.EventDay\n"+
                "GROUP BY CONVENTIONDAY.WeekDay\n"+
                "ORDER BY CONVENTIONDAY.WeekDay");

        if (result != null) {
                System.out.println("17. A list of all events happening at the
convention, where they're happening, and when they're happening, sorted by where
they're happening, then by when.");
                System.out.println("Day\tNumber of Events");
                    while(result.next()) {
                        System.out.println(result.getString("Day")+ "\t" +
result.getString("Number of Events"));
                    }
                }
        System.out.println();

        result = connection.execute("SELECT EventID AS \"Event IDs\",
COUNT(EmployeeID) AS \"Number of Employees\"\n" +
                    "FROM WORKSAT\n" +
                    "GROUP BY EventID\n" +
                    "ORDER BY COUNT(EmployeeID) DESC");
        if (result != null) {
            System.out.println("18. Show how many employees are working at each
event");
            System.out.println("Event IDs\tNumber of Employees");
            while(result.next()) {
                System.out.println(result.getString("Event
IDs")+"\t\t"+result.getString("Number of Employees"));
            }
        }
        System.out.println();

        result = connection.execute("SELECT ev.EventDay, e.EmployeeID,
e.EmployeeName\n"+
                "FROM employee e, event ev\n"+
                "WHERE EXISTS\n"+
                "(SELECT * FROM worksat w\n"+
                    "WHERE w.EventID = ev.EventID\n"+
                        "AND w.employeeID = e.employeeID\n"+
                        "AND ev.EventDay = 1\n"+
                ")\n"+
                "UNION\n"+
                "(SELECT ev.EventDay, e.EmployeeID, e.EmployeeName\n"+
                "FROM employee e, event ev\n"+
                "WHERE EXISTS\n"+
                "(SELECT * FROM worksat w\n"+
                    "WHERE w.EventID = ev.EventID\n"+
                        "AND w.employeeID = e.employeeID\n"+
                        "AND ev.EventDay = 2\n"+
                ")\n"+
                ")");
```

```
            if (result != null) {
                System.out.println("19. A list of all employees working days 1 and
2.");
                System.out.println("EventDay\tEmployeeID\tEmployeeName");
                    while(result.next()) {
                        System.out.println(result.getString("EventDay")+ "\t\t" +
result.getString("EmployeeID")+"\t\t" + result.getString("EmployeeName"));
                    }
                }
            System.out.println();

            result = connection.execute("SELECT EVENT.EventName AS \"Event\",
COUNT(EMPLOYEE.EmployeeID) AS \"Number of Employees\"\n" +
                    "FROM EVENT\n" +
                    "INNER JOIN WORKSAT\n" +
                    "ON EVENT.EventID = WORKSAT.EventID\n" +
                    "INNER JOIN EMPLOYEE\n" +
                    "ON WORKSAT.EmployeeID = EMPLOYEE.EmployeeID\n" +
                    "GROUP BY Event.EventName\n" +
                    "HAVING COUNT(EMPLOYEE.EmployeeID) > 1\n" +
                    "ORDER BY COUNT(EMPLOYEE.EmployeeID) DESC");
            if (result != null) {
                System.out.println("20. list of all events happening at the
convention and their details");
                System.out.println("Event\t\t\t\tNumber of Employees");
                while(result.next()) {

System.out.println(result.getString("Event")+"\t"+result.getString("Number of
Employees"));
                }
            }
            System.out.println();
      }
}
```

Conclusion

Being able to work on a DBMS throughout the year has been a very pleasant experience. Each assignment in this project has allowed us to better understand the concepts and ideas learned in class through hands-on experience. The assignments were also spaced out and distributed in a way that prevented heavy strain on any of the group members, which is a small but rather important note as a submission was expected every week. Had these assignments been more content-heavy, our ability to learn in each assignment would have been significantly lowered. We found this project so enjoyable because it gave us an opportunity to see how a database could be designed for a real application. Beginning with conceptual design like ER diagrams, getting a chance to actually implement it using SQL, and further refining it using normal forms were all experiences we think will be very useful if we choose to pursue a career in database management or design.