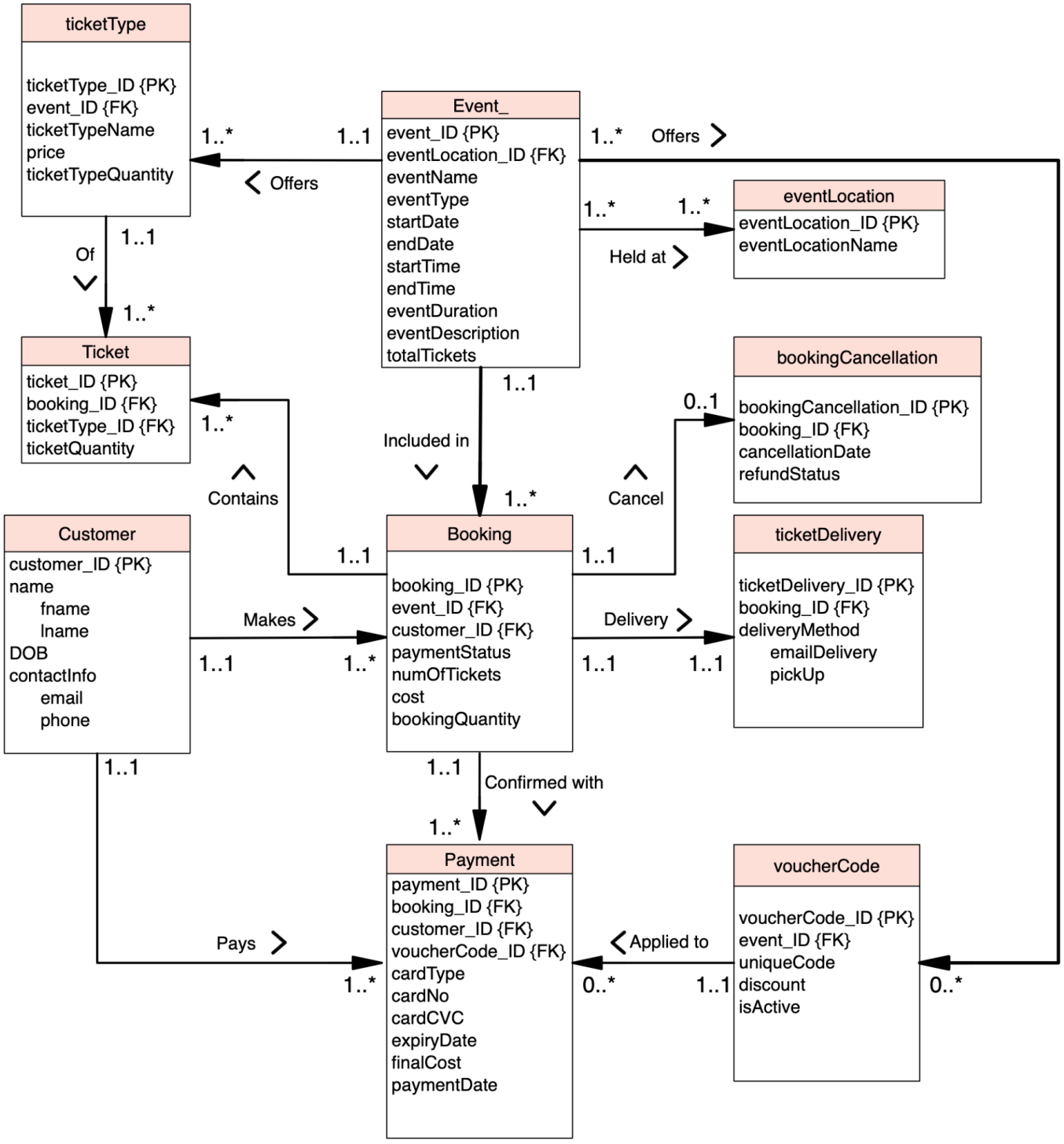
# Ticket Booking Database Design Report

## Conceptual model design



## Entities

1. **Customer** : customer_ID, fname, lname, DOB, email, phone
    - ➢ A customer will make a booking of tickets of particular types affiliated with specific events.
    - ➢ The customer_ID, along with all other IDs will be of type ‘VARCHAR’ to make the IDs easily distinguishable with letters and numbers. For example, customer 1 will have the ID ‘C1’.
    - ➢ I also made the ‘phone’ attribute of type ‘VARCHAR’ due to the leading ‘0’ of a phone number such as ‘07753423128’.
2. **eventLocation :** eventLocation_ID, eventLocationName
    - ➢ This entity stores where the event is held. The ‘eventLocation_ID’ primary key is the postcode of the event to make the location ID distinguishable.
3. **Event_** : event_ID, eventLocation_ID, eventName, eventType, startDate, endDate, startTime, endTime, eventDuration, eventDescription, totalTickets
    - ➢ An event will have a type such as ‘Music’. The ‘totalTickets’ attribute stores the total tickets for a given event, which is the sum of the ‘ticketTypeQuantity’ values for each ticket type of a single event.
    - ➢ I added an underscore to avoid confusion with the ‘EVENT’ Scheduler in SQL.
    - ➢ The ‘eventDuration’ attribute represents the length of the event in hours.
4. **Booking :** booking_ID, event_ID, customer_ID, paymentStatus, numOfTickets, cost, bookingQuantity
    - ➢ A booking contains tickets, and is directly linked to the customer via ‘customer_ID’ as its foreign key. One booking can only be for one event hence the ‘event_ID’ foreign key.
    - ➢ ‘bookingQuantity’ allows for a customer to have multiple bookings of the same ticket type and quantity (duplicate). If not the same tickets, then that will be stored in a separate booking.
    - ➢ ‘paymentStatus’ will have the value ‘1’ or ‘0’ meaning paid or not paid.
5. **bookingCancellation** : bookingCancellation_ID, booking_ID, cancellationDate, refundStatus
    - ➢ Each cancellation of a booking has its own ID and the entity has ‘booking_ID’ as its foreign key to refer to the booking being cancelled.
    - ➢ The ‘refundStatus’ attribute will have the values ‘1’ refunded or ‘0’ not refunded.
6. **voucherCode** : voucherCode_ID, event_ID, uniqueCode, discount, isActive
    - ➢ Each voucher code is affiliated with an event, hence the ’event_ID’ foreign key. The ‘uniqueCode’ is the code that a customer uses to discount their payment.
    - ➢ The ‘isActive’ attribute will have values: ‘1’ is active or ‘0’ inactive.

- ➢ The ‘discount’ attribute will be in decimal form.

7. **Payment** : payment_ID, booking_ID, customer_ID, voucherCode_ID, cardType, cardNo, cardCVC, expiryDate, finalCost, paymentDate
   - ➢ Each payment is linked to a customer and a booking via foreign keys ‘booking_ID’ and ‘customer_ID’.
   - ➢ If a payment has been discounted with a voucher code, it will include the voucher code ID as a foreign key and the ‘finalCost’ attribute contains the booking cost after applying a discount if one is present.
8. **Ticket** : ticket_ID, booking_ID, ticketType_ID, ticketQuantity
   - ➢ The ‘Ticket’ entity contains the details of each ticket within a booking so it has ‘booking_ID’ as its foreign key.
   - ➢ Each ticket has a type distinguished by the foreign key ‘ticketType_ID’
   - ➢ The ‘ticketQuantity’ attribute indicates the quantity of tickets of a given type included in a booking.
9. **ticketType** : ticketType_ID, event_ID, ticketTypeName, price, ticketTypeQuantity
   - ➢ Each ticket type has a price and is linked to an event via the ‘event_ID’ foreign key.
   - ➢ The ‘ticketTypeQuantity’ attribute stores the count of a specific ticket type that an event offers.
10. **ticketDelivery** : ticketDelivery_ID, booking_ID, emailDelivery, pickUp
    - ➢ Ticket delivery is linked to a booking via the ‘booking_ID’ foreign key.
    - ➢ ‘emailDelivery’ and ‘pickUp’ have values ‘1’ true or ‘0’ false.

## Relations

- **Customer → Booking** (1..1 → 1..*) Each customer can have 1 or many bookings, a customer does not exist without a booking.
- **Customer → Payment** (1..1 → 1..*) Each customer is associated with 1 or many payments.
- **Booking → Payment** (1..1 → 1..*) There can be 1 or many payments for a single booking (the cost of a booking can be subdivided into several payments).
- **Booking → Ticket** (1..1 → 1..*) A booking can include 1 or many tickets.
- **Event_ → Booking** (1..1 → 1..*) There can be 1 or many bookings for a single event. 1 booking cannot include several events.
- **Booking → ticketDelivery** (1..1 → 1..1) Each single booking can only have 1 ticketDelivery.
- **Booking → bookingCanellation** (1..1 → 0..1) A booking can have either no cancellation or 1 cancellation.
- **Event_ → ticketType** (1..1 → 1..*) For a single event, there are 1 or many types of ticket.

- **Event_ → eventLocation** (1..* → 1..*) An event can have 1 or many locations, many events can be held at a single location.
- **Event_ → voucherCode** (1..* → 0..*) Many events can have the same or different voucher codes, 1 or many events can have no voucher codes affiliated with them.
- **ticketType → Ticket** (1..1 → 1..*) Each single ticket can only have 1 type, however, several tickets can have the same type.
- **voucherCode → Payment** (1..1 → 0..*) A single payment can only have 1 voucherCode applied to it and many payments can have the same voucher code or no voucher codes applied.

## Relational Model

Customer(customer_ID,fname,lname,DOB,email,phone)
**Primary key(s):** customer_ID

eventLocation(eventLocation_ID,eventLocationName)
**Primary key:** eventLocation_ID

Event_(event_ID,eventLocation_ID,eventName,eventType,startDate,endDate,startTime, endTime,eventDuration,eventDescription,totalTickets)
**Primary key:** event_ID
**Foreign key(s):** eventLocation_ID references eventLocation(eventLocation_ID)

Booking(booking_ID,event_ID,customer_ID,paymentStatus,numOfTickets,cost, bookingQuantity)
**Primary key:** booking_ID
**Foreign key(s):** event_ID references Event_(event_ID)
customer_ID references Customer(customer_ID)

bookingCancellation(bookingCancellation_ID,booking_ID,cancellationDate, refundStatus)
**Primary key:** bookingCancellation_ID
**Foreign key(s):** booking_ID references Booking(booking_ID)

voucherCode(voucherCode_ID,event_ID,uniqueCode,discount,isActive)
**Primary key:** voucherCode_ID
**Foreign key(s):** event_ID references Event_(event_ID)

Payment(payment_ID,booking_ID,customer_ID,voucherCode_ID,cardType,cardNo,card CVC,expiryDate,finalCost,paymentDate)
**Primary key:** payment_ID

**Foreign key(s):** booking_ID references Booking(booking_ID)
customer_ID references Customer(customer_ID)
voucherCode_ID references voucherCode(voucherCode_ID)

ticketDelivery(ticketDelivery_ID,booking_ID,emailDelivery,pickUp)
**Primary key:** ticketDelivery_ID
**Foreign key(s):** booking_ID references Booking(booking_ID)

ticketType(ticketType_ID,event_ID,ticketTypeName,price,ticketTypeQuantity)

**Primary key:** ticketType_ID
**Foreign key(s):** event_ID references Event_(event_ID)

Ticket(ticket_ID,booking_ID,ticketType_ID,ticketQuantity)
**Primary key:** ticket_ID
**Foreign key(s):** booking_ID references Booking(booking_ID)
ticketType_ID references ticketType(ticketType_ID)