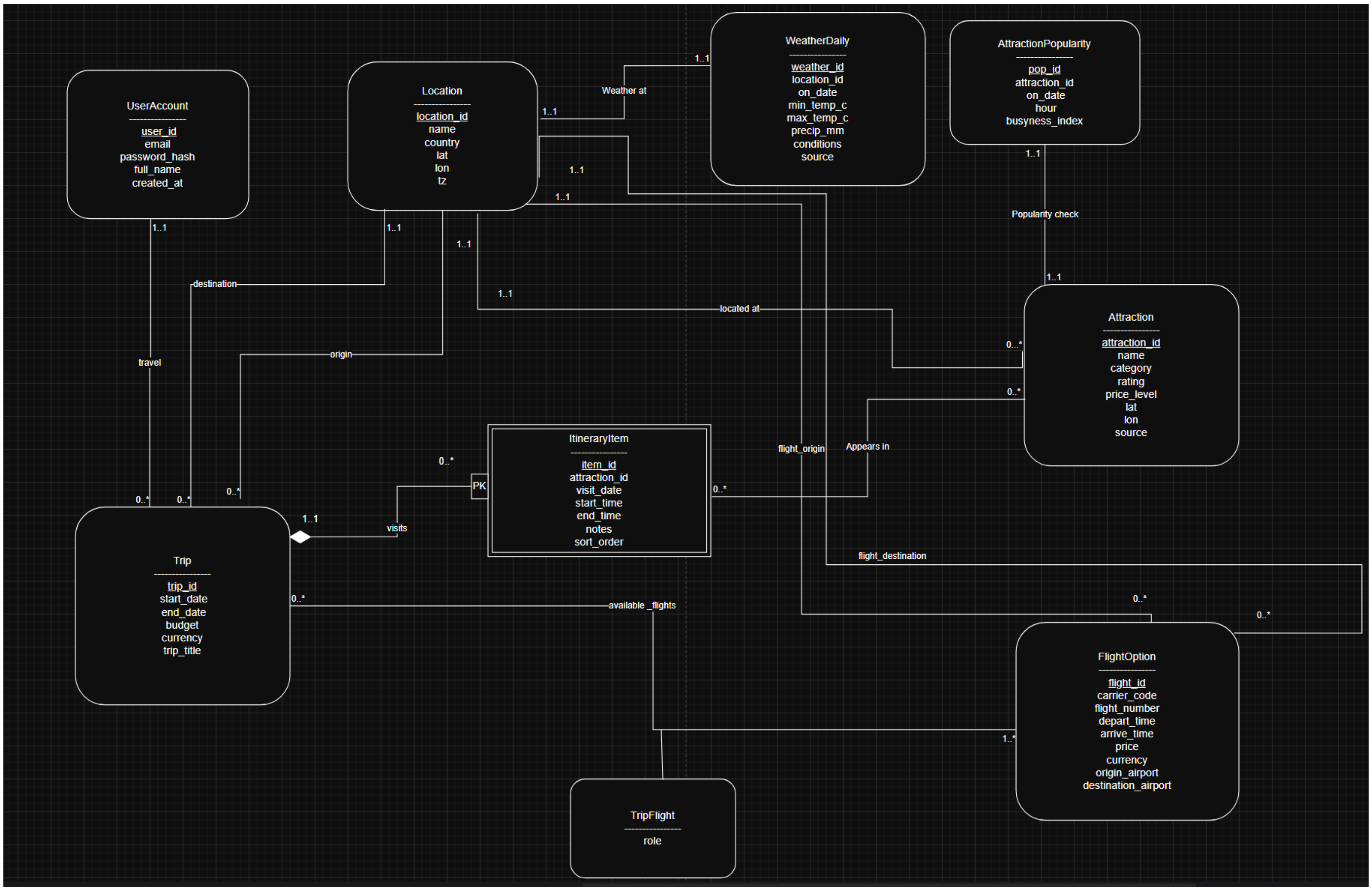
Stage 2
We are creating a UML Diagram.



UML Diagram:

Explain your assumptions for each entity and relationship in your model. Discuss why you've modeled something as an entity rather than an attribute of another entity. Describe the cardinality of relationships, like why a student is linked to only one advisor. These assumptions might come from customer requirements or application constraints. Please clarify them.

- User Account
  - The user account is an independent entity that represents a user of a program. The user contains identifiable information such as the user_id, email, password, full name, and creation date. It is an independent entity because users can exist without having any trips, and each user can have multiple trips (0..*). Because of

    - the above, it has a 1-to-many relationship with trips, and each trip can only be created by one user.
- Trip
    - The trip is an entity that represents a single user outing. It is modeled as an entity because it is the foundation of a trip, containing the information about that one trip. It is independent of the user because a user can have multiple trips. It contains identifiers such as trip id and user id. Trip descriptions include origin_location_id, destination_location_id, start_date, end_date, budget, currency, trip_title. A trip must have a user who created it (1..1). It also must have an origin and destination location (1..1), and some flights that connect the two locations (1..*). It can have none or many itinerary items (0..*) as that can be modified easily and on a whim.
- Trip Flight
    - Trip flight is an identity because it is an organization of the Flight Options table specific to any trip. This is useful as it can contain many flights in order to get from the origin location to the destination location. Therefore, it can be connected to any trip (0..*) or no trip at all because it is a suggestion. It can also be connected to multiple flights (1..*) as one flight is needed to get somewhere, but multiple flights might be needed to get to the specific place that you are looking for.
- Flight Options
    - Flight options is an entity because the same flight can be suggested across many users/trips. It also serves as a cache and provides reuse for the options pulled from APIs/datasets. The number of attributes specific to the flight such as the flight id, carrier code, flight number, origin and destination id, depart and arrive times, price, and currency, justifies a standalone table to store API information.
- Location
    - The Location entity represents a geographic area such as a city or region, containing stable information like name, country, latitude, longitude, and time zone. It serves as the foundation for weather data, attractions, and flights. Modeling it as a separate entity rather than as attributes of other tables avoids redundancy and allows multiple datasets, like weather, attractions, and trips, to connect to a single location. Each location can have many weather records and attractions, but is uniquely identified in trip origin and destination relationships.
- Weather Daily
    - WeatherDaily stores daily environmental data for each location, including minimum and maximum temperature, precipitation, and conditions. This information changes daily, so it’s modeled as its own entity instead of being part of Location. Each WeatherDaily record is tied to exactly one location, but a single location can have many weather entries over time. The relationship’s one-to-many

structure reflects that a city's weather evolves across different dates, and storing it separately allows for historical queries and data visualization.

- Itinerary Item
    - The ItineraryItem entity represents an individual event or activity within a user's trip. It connects a trip to a specific attraction while storing extra details such as visit date, start and end time, notes, and sort order. It's modeled as an entity rather than a relationship because it carries meaningful attributes beyond just linking a trip and attraction. Each trip can include many itinerary items, but every itinerary item belongs to exactly one trip and references a single attraction. This design allows for flexible and detailed scheduling within a trip.
- Attraction
    - The Attraction entity defines points of interest such as landmarks, museums, or restaurants, with attributes like name, category, rating, and price level. It's linked to one Location since each attraction is tied to a specific geographic area. Making attractions a standalone entity allows them to be shared across multiple users and trips, making it possible to aggregate data like popularity trends or recommendations. The one-to-many relationship from Location to Attraction captures that a city typically has many attractions.
- Attraction Popularity
    - AttractionPopularity captures the crowd or busyness level for a particular attraction at a given hour and date. It changes frequently and represents time-based data, which is why it's modeled as its own entity rather than an attribute of Attraction. Each record measures how busy an attraction is during a specific time, and each attraction can have many popularity readings. The one-to-many relationship allows the system to store a full timeline of popularity metrics for crowd prediction and recommendations on when to visit.

Relational Schema

Location(location_id:INT [PK], name:VARCHAR(160), country:VARCHAR(80), lat:DECIMAL(9,6), lon:DECIMAL(9,6), tz:VARCHAR(40))

(Preity)Destination(trip_id:INT [PK, FK to Trip.trip_id], location_id:INT [FK to Location.location_id])
(Preity)Origin(trip_id:INT [PK, FK to Trip.trip_id], location_id:INT [FK to Location.location_id])

The destination and origin tables are the 1 to many relationship set between Location and trip.

flight_origin(flight_id:INT [PK, FK to FlightOption.flight_id], location_id:INT [FK to Location.location_id])
flight_destination(flight_id:INT [PK, FK to FlightOption.flight_id], location_id:INT [FK to Location.location_id])

We do the same thing with flight destination and origin since these are also 1 to many relations.

WeatherDaily(weather_id:INT [PK], location_id:INT [FK to Location.location_id], on_date:DATE, min_temp_c:DECIMAL(5,2), max_temp_c:DECIMAL(5,2), precip_mm:DECIMAL(6,2), conditions:VARCHAR(64), source:VARCHAR(40))

(Kevin)Attraction(attraction_id:INT [PK], location_id:INT [FK to Location.location_id], name:VARCHAR(200), category:VARCHAR(40), rating:DECIMAL(2,1), price_level:TINYINT, lat:DECIMAL(9,6), lon:DECIMAL(9,6), source:VARCHAR(40))

(Kevin)AttractionPopularity(pop_id:INT [PK], attraction_id:INT [FK to Attraction.attraction_id], on_date:DATE, hour:TINYINT, busyness_index:TINYINT)

(Kevin)ItineraryItem(item_id:INT, trip_id:INT [FK to Trip.trip_id], attraction_id:INT [FK to Attraction.attraction_id], visit_date:DATE, start_time:TIME, end_time:TIME, notes:VARCHAR(400), sort_order:INT, (item_id,trip_id)[PK])
For the itinerary item, we have item_id and trip_id together as the primary key b/c the itinerary item is not identifiable without trip_id. It is a weak entity of Trip.

Orlando
UserAccount(user_id:INT [PK], email: VARCHAR(255) [UNIQUE], password_hash: VARCHAR(255), full_name:VARCHAR(120), created_at: TIMESTAMP)

travel(trip_id:INT[PK] , user_id:INT)

Trip(trip_id: INT [PK] , start_date: DATE , end_date: DATE , budget:DECIMAL(10,2) , currency:VARCHAR(3) , trip_title:VARCHAR(120))

FlightOption (flight_id : INT [PK] , carrier_code : VARCHAR(12) , flight_number: VARCHAR(12) , depart_time : TIMESTAMP , arrive_time: TIMESTAMP , price : DECIMAL(10,2) , currency: VARCHAR(3))

(Preity)TripFlight(trip_id:INT [FK to Trip.trip_id], flight_id : INT [FK to FlightOption.flight_id] ,role: VARCHAR(16), (trip_id , flight_id)[PK])

BCNF Check

- UserAccount –
    - Every non-key attribute (email, password_hash, full_name, created_at) depends only on user_id, the primary key.
- Location –
    - Attributes (name, country, lat, lon, tz) depend solely on location_id.
- Trip –
    - Each trip is identified by trip_id; all other attributes (start_date, end_date, budget, currency, trip_title) depend entirely on that key.
- Origin / Destination –
    - Each table's key (trip_id) uniquely determines the associated location_id.
- FlightOption –
    - flight_id uniquely determines all flight details (carrier_code, flight_number, depart_time, etc.).
- flight_origin / flight_destination –
    - Each flight_id determines exactly one location_id.
- TripFlight –
    - Composite key (trip_id, flight_id) uniquely identifies each record and determines the role.
- WeatherDaily –
    - weather_id is the primary key, and all attributes (location_id, on_date, min_temp_c, etc.) depend only on it.
- Attraction –
    - All descriptive attributes depend solely on attraction_id.
- AttractionPopularity –
    - pop_id uniquely identifies each record; attraction_id, on_date, and hour are dependent on that key only.
- ItineraryItem –
    - The composite key (item_id, trip_id) determines all other attributes (attraction_id, visit_date, start_time, etc.).
- travel –
    - trip_id is both the primary key and a foreign key to Trip.trip_id; it determines exactly one user_id.