SQL Session 2

1. Column renaming in query result and using AND

Renaming an output column

```
SELECT name, ssn AS travler_ssn
FROM Traveler;
```

Chaining boolean expressions

```
SELECT name, ssn AS traveler_ssn
FROM Traveler
WHERE name NOT LIKE "Mike%"
AND name NOT LIKE "%White";
```

2. NULL

Insert an Agent with no phone number or years_experience

```
INSERT INTO TravelAgent (name)
VALUES ("Ned Nullman Nullster");
```

Get all agents

```
SELECT *
FROM TravelAgent;
```

Three valued logic causes this boolean expression (That should be true) be Unknown for Ned Nullster

```
SELECT *
FROM TravelAgent
WHERE years_experience > 10
OR years_experience <= 10;
```

3. Multi-table Queries

Notice that selecting from both tables at once results in the cartesian product

```
SELECT *
FROM Booking, Traveler;
```

We need a condition to only get Booking and Traveler rows that correspond with each other, this is done by checking the PK

```
SELECT *
FROM Booking, Traveler
WHERE traveler_ssn = ssn;
```

This is somewhat personal preference, but I prefer to use this method to keep track of tables and their attributes

```
SELECT *
FROM Booking B, Traveler T
WHERE B.traveler_ssn = T.ssn;
```

Getting Travelers that had trips booked by an agent with the first

name Alice

```
SELECT *
FROM Booking B, Traveler T
WHERE B.traveler_ssn = T.ssn
AND agent LIKE "Alice%";
```

4. Self Joins

Find trips that start where another one ends

```
SELECT *
FROM Trip T1, Trip T2
WHERE T1.start_location = T2.end_location;
```

Get just the trips with qualifying starting locations

(Note the T1.* syntax)

```
SELECT T1.*
FROM Trip T1, Trip T2
WHERE T1.start_location = T2.end_location;
```

Find the id and locations for trips that share a start and end location

```
SELECT T1.id as T1_ID, T1.start_location as T1_Start, T1.end_location as T1_
T2.id as T2_ID, T2.start_location as T2_Start, T2.end_location as T2_End
FROM Trip T1, Trip T2
WHERE T1.start_location = T2.end_location;
```

5. Distinct

List all start locations

```
SELECT start_location
FROM Trip;
```

List all start locations but without the duplicates

```
SELECT distinct start_location
FROM Trip;
```

6. IN and NOT IN

Get All Travelers

```
SELECT *
FROM Traveler;
```

Get all travelers that have an odd SSN in the range 101-109

```
SELECT *
FROM Traveler
WHERE ssn IN (101, 103, 105, 107, 109);
```

Get all travelers that we didn't get above

```
SELECT *
FROM Traveler
WHERE ssn NOT IN (101, 103, 105, 107, 109);
```

7. Subqueries

Get all travelers that have been to Paris

```
SELECT *
FROM Traveler
WHERE ssn IN (
    SELECT B.traveler_ssn
    FROM Booking B, Trip T
    WHERE B.trip_id = T.id
    AND end_location = "Paris");
```

> ### Find bookings and trips where the trip ends in paris
>
> ```
> SELECT *
> FROM Booking B, Trip T
> WHERE B.trip_id = T.id and end_location =
> "Paris";
> ```

8. Exists

Alternate way to find travelers that have been to Paris

```
SELECT *
FROM Traveler
WHERE EXISTS (
    SELECT *
    FROM Booking B, Trip T
    WHERE B.trip_id = T.id and end_location = "Paris" and B.traveler_ssn =
```

9. ANY and ALL

Get all Travelers whose name end in 'n'

```
SELECT *
FROM Traveler T
WHERE name like "%n";
```

Get all travelers who are younger than all travelers whose last name ends in 'n'

```
SELECT *
FROM Traveler
WHERE dob < ALL (
    SELECT T_Sub.dob
    FROM Traveler T_Sub
    WHERE name like "%n"
);
```

> Saying "Get all travelers who are younger than all travelers whose last name ends in 'n' " is like saying "Get all travelers who are younger than the youngest travelers whose last name ends in 'n' "
>
> Less than all x = Less than MIN(x)
> More than all x = More than MAX(x)
> Less than any x = Less than MAX(x)
> More than any x = More than MIN(x)
>
> ```
> SELECT *
> FROM Traveler
> WHERE dob < (
>     SELECT min(T_Sub.dob)
>     FROM Traveler T_Sub
>     WHERE name like "%n"
> );
> ```

10. Set Operations

Get all travelers who have been on a trip

```
(SELECT ssn FROM Traveler)
INTERSECT
(SELECT traveler_ssn FROM Booking);
```

11. Joins

Get all bookings and their corresponding agent

```
SELECT *
FROM Booking
JOIN TravelAgent ON agent=name;
```

Get all agents and any corresponding bookings

```
SELECT *
FROM Booking
RIGHT OUTER JOIN TravelAgent ON agent = name;
```

12. Group By and Aggregates

Get all trips

```
SELECT *
FROM Trip;
```

Get how many times a start location has been used

```
SELECT \*, count(*)
FROM Trip
GROUP BY start_location;
```

Get all locations where more than one trip have started from

```
SELECT start_location, count(\*)
FROM Trip
GROUP BY start_location
HAVING count(*) > 1;
```

Get the average trip start date for a location

```
SELECT start_location, avg(start_date)
FROM Trip
GROUP BY start_location;
```

13. Query Composition Exercise

Query 1. Find the name and years of experience for the agent with the most experience

Query 2. Find the names of all travelers who have gone on more than one trip

Query 3. Find travelers who have traveled together

Query 4. Find the names of agents who booked trips to Rome for travelers born after 1997

Query 5. Get all names in the database

Query 6. Get the total days travelled for each traveler

14. Answers

Query 1.

Option 1: Using MAX

```
SELECT name, years_experience
FROM TravelAgent
WHERE years_experience = (
    SELECT MAX(years_experience)
    FROM TravelAgent
);
```

Option 2: Not Using MAX

```
SELECT name, years_experience
FROM TravelAgent
WHERE years_experience NOT IN (
    SELECT distinct T1.years_experience
    FROM TravelAgent T1, TravelAgent T2
    WHERE T1.years_experience < T2.years_experience
);
```

> #### Get years_experience other than the greatest value
>
> ```
> SELECT distinct T1.years_experience
> FROM TravelAgent T1, TravelAgent T2
> WHERE T1.years_experience < T2.years_experience;
> ```

Query 2.

Option 1: Using Join and HAVING

```
SELECT name
FROM Booking
JOIN Traveler ON traveler_ssn = ssn
GROUP BY traveler_ssn
HAVING count(*) > 1;
```

Option 2: Using IN and a Subquery

```
SELECT name
FROM Traveler
WHERE ssn in (
    SELECT traveler_ssn
    FROM Booking
    GROUP BY traveler_ssn
    HAVING count(*) > 1
);
```

Option 3: Just using a Subquery

```
SELECT name
FROM Traveler
WHERE 1 < (
    SELECT count(*)
    FROM Booking
    WHERE traveler_ssn = ssn
    GROUP BY traveler_ssn
);
```

Query 3.

Option 1: Self joining

```
SELECT T1.name as Traveler_1, T2.name as Traveler_2
FROM Booking B1 JOIN Traveler T1 ON B1.traveler_ssn = T1.ssn,
Booking B2 JOIN Traveler T2 ON B2.traveler_ssn = T2.ssn
WHERE B1.trip_id = B2.trip_id
AND B1.traveler_ssn < B2.traveler_ssn;  -- We use < instead of <> to remove
-- Try <> and see what happens
```

> **Get all qualifying bookings**
>
> ```
> SELECT *
> FROM Booking B1, Booking B2
> WHERE B1.trip_id = B2.trip_id AND B1.traveler_ssn < B2.traveler_ssn;
> ```

Option 2: Decreasing complexity with a subquery

```
SELECT T1.name as Traveler_1, T2.name as Traveler_2
FROM Traveler T1, Traveler T2
WHERE (T1.ssn, T2.ssn) IN (
    SELECT B1.traveler_ssn, B2.traveler_ssn
    FROM Booking B1, Booking B2
    WHERE B1.trip_id = B2.trip_id
    AND B1.traveler_ssn < B2.traveler_ssn
);
```

Query 4.

```
SELECT agent
FROM Booking
JOIN Traveler ON traveler_ssn = ssn
JOIN Trip on trip_id = id
WHERE end_location = "Rome"
AND dob >= "1998-01-01";
```

Query 5.

```
(SELECT name FROM Traveler)
UNION
(SELECT name FROM TravelAgent);
```

Query 6.

```
SELECT T.*, sum(end_date - start_date) as TotalDaysTravelling
FROM Trip
JOIN Booking B on trip_id = id
JOIN Traveler T on traveler_ssn = ssn
GROUP BY ssn;
```

Misc query to get all data

```
SELECT B.trip_id AS "B.TripID",
B.traveler_ssn AS "B.TravelerSSN",
B.agent AS "B.Agent",
Trip.id AS "Trip.ID",
Trip.start_date AS "Trip.StartDate",
Trip.end_date AS "Trip.EndDate",
Trip.start_location AS "Trip.StartLoc",
Trip.end_location AS "Trip.EndLoc",
T.ssn AS "T.SSN",
T.name AS "T.Name",
T.dob AS "T.DOB",
TA.name AS "TA.Name",
TA.phone AS "TA.Phone",
TA.years_experience AS "TA.YearsExp"
FROM Trip
JOIN Booking B ON trip_id = id
JOIN Traveler T ON traveler_ssn = ssn
JOIN TravelAgent TA ON B.agent = TA.name;
```