Pavel Kuznetsov
CS-1

Database Project

Description of the information system

The entities involved are: Experience, Navigation Type, Road Type, Maneuver Type, Traffic Type, and Weather Type.

Information System Rules

An experience can belong to only one navigation type.
A navigation type can belong to zero or multiple experiences.

An experience can belong to one or multiple road types.
A road type can belong to zero or multiple experiences.

An experience can have zero or multiple maneuver types.
A maneuver type can belong to zero or multiple experiences.

An experience can belong to one or multiple traffic types.
A traffic type can belong to zero or multiple experiences.

An experience can happen during one or multiple weather types.
A weather type can belong to zero or multiple experiences.

Conceptual Data Model

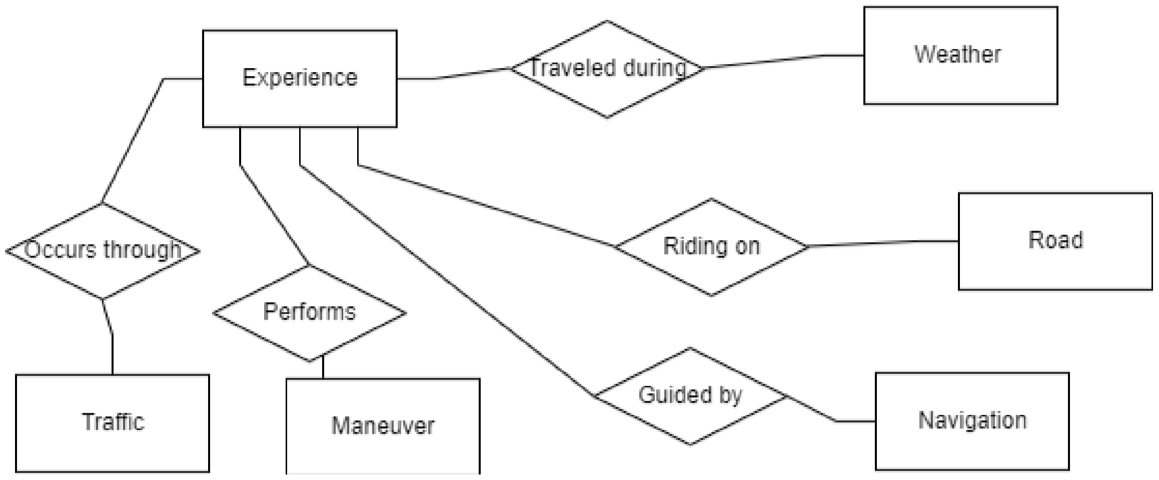
[Experience](Traveled during)[Weather]

[Experience](Riding on)[Road]

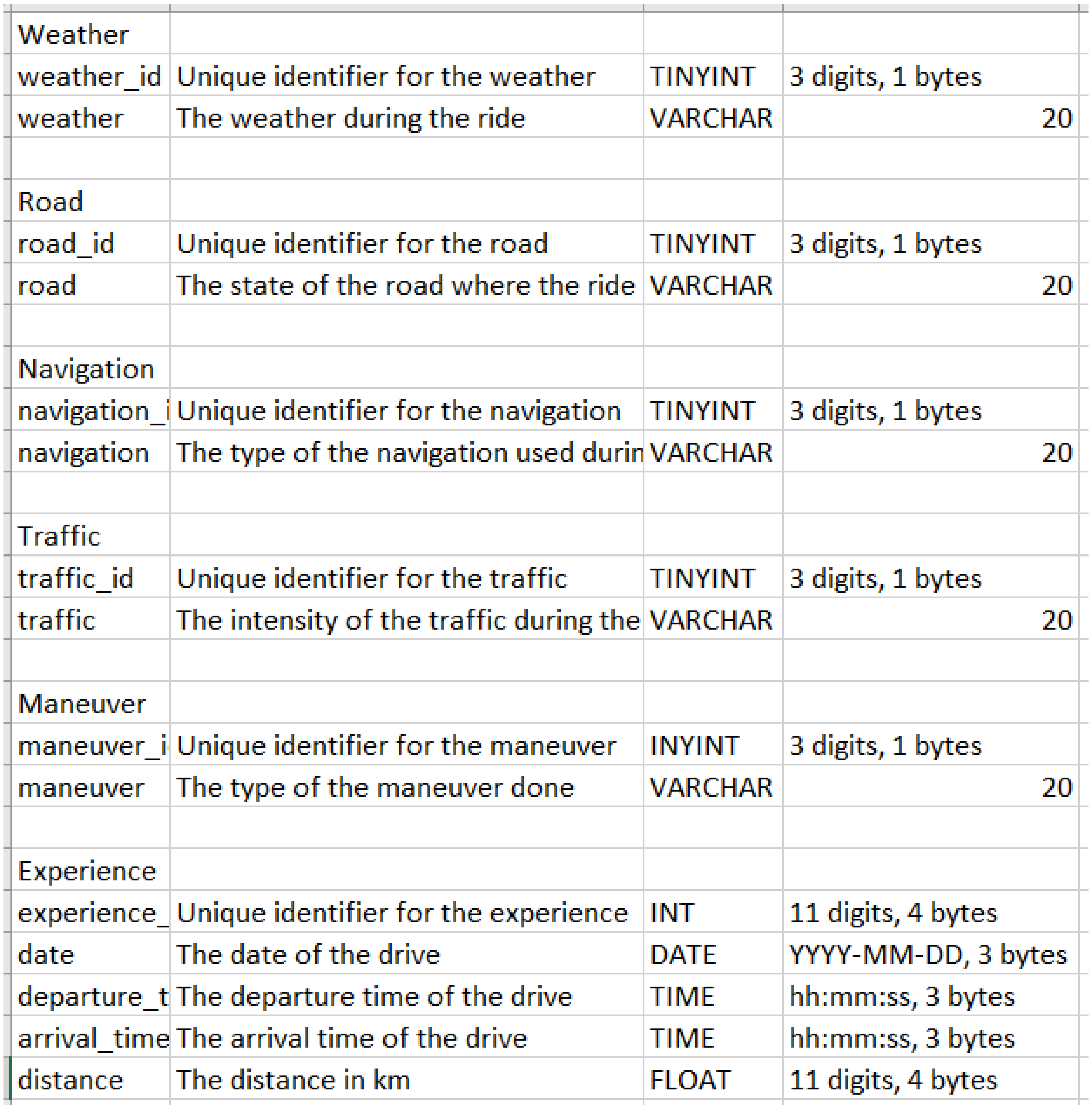
[Experience](Guided by)[Navigation]

[Experience](Occurs through)[Traffic]

[Experience](Performs)[Maneuver]



Data Dictionary



Weather			
weather_id	Unique identifier for the weather	TINYINT	3 digits, 1 bytes
weather	The weather during the ride	VARCHAR	20
Road			
road_id	Unique identifier for the road	TINYINT	3 digits, 1 bytes
road	The state of the road where the ride	VARCHAR	20
Navigation			
navigation_i	Unique identifier for the navigation	TINYINT	3 digits, 1 bytes
navigation	The type of the navigation used durin	VARCHAR	20
Traffic			
traffic_id	Unique identifier for the traffic	TINYINT	3 digits, 1 bytes
traffic	The intensity of the traffic during the	VARCHAR	20
Maneuver			
maneuver_i	Unique identifier for the maneuver	INYINT	3 digits, 1 bytes
maneuver	The type of the maneuver done	VARCHAR	20
Experience			
experience_	Unique identifier for the experience	INT	11 digits, 4 bytes
date	The date of the drive	DATE	YYYY-MM-DD, 3 bytes
departure_t	The departure time of the drive	TIME	hh:mm:ss, 3 bytes
arrival_time	The arrival time of the drive	TIME	hh:mm:ss, 3 bytes
distance	The distance in km	FLOAT	11 digits, 4 bytes

Logical Data Model

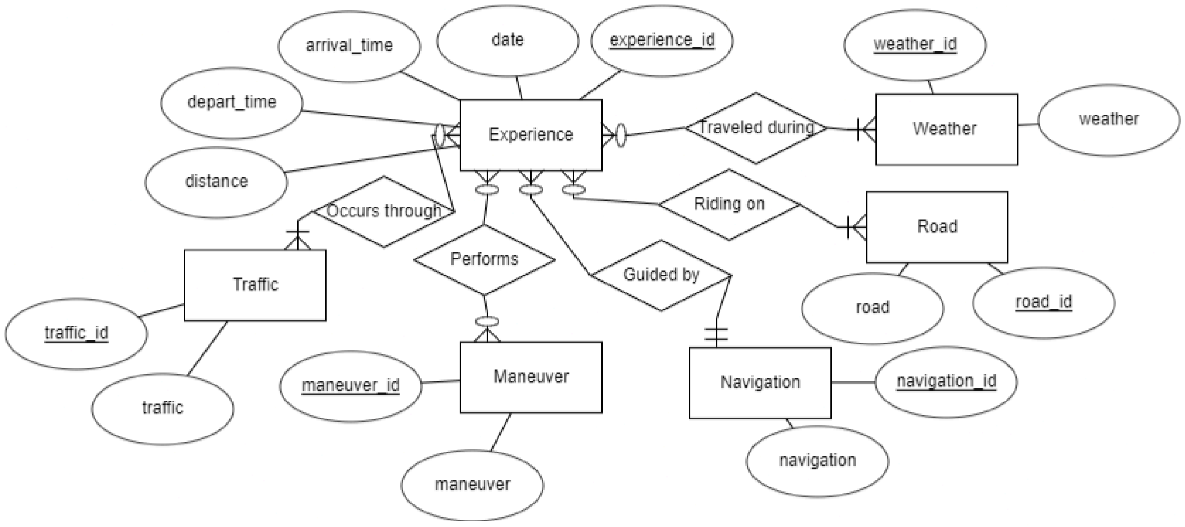
1. Navigation Type and Experience:
- Relationship Type: One-to-Many
Each experience can belong to only one navigation type, and each navigation type can belong to zero or multiple experiences.

2. Road Type and Experience:
-Relationship Type: Many-to-Many
Each experience can belong to one or multiple road types, and each road type can belong to zero or multiple experiences.

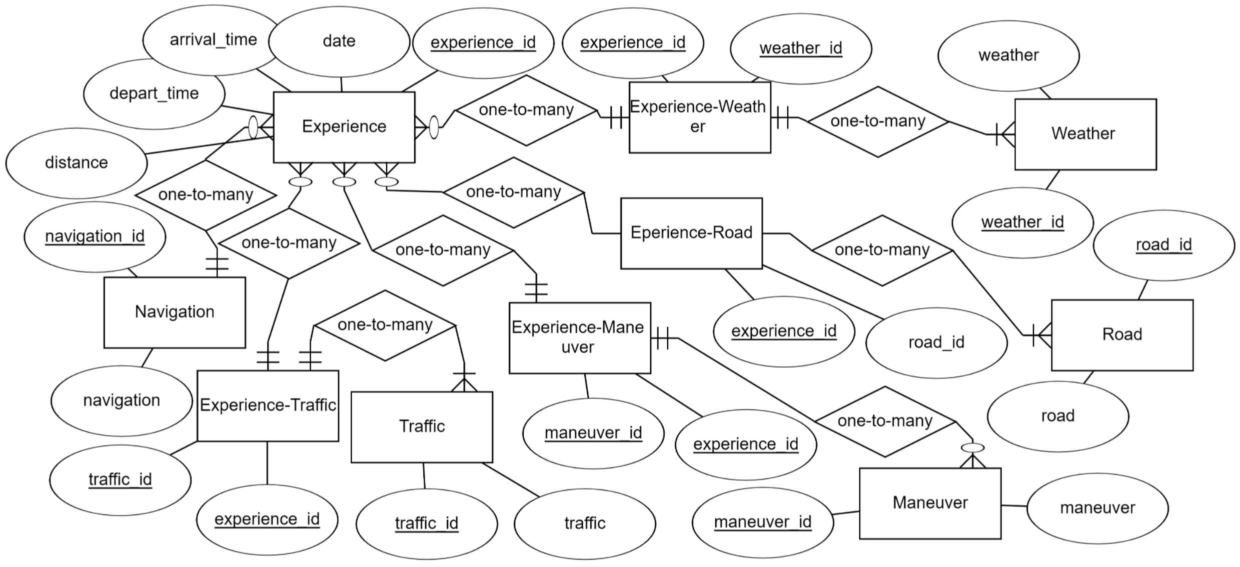
3. Maneuver Type and Experience:
- Relationship Type: Many-to-Many
Each experience can have zero or multiple maneuver types, and each maneuver type can belong to zero or multiple experiences.

4. Traffic Type and Experience:
- Relationship Type: Many-to-Many
Each experience can belong to one or multiple traffic types, and each traffic type can belong to zero or multiple experiences.

5. Weather Type and Experience:
Relationship Type: Many-to-Many
Each experience can happen during one or multiple weather types, and each weather type can belong to zero or multiple experiences.



Physical Data Model



Experience:
- Primary Key (PK): experience_id
- Foreign Key (FK): navigation_id
- Description: The Experience entity represents individual experiences and is uniquely identified by experience_id. It includes a foreign key navigation_id to establish a relationship with the Navigation entity.

Experience_Traffic:
- Foreign Keys (FK): experience_id, traffic_id
- Description: The Experience_Traffic entity is a junction table that links Experience and Traffic entities in a many-to-many relationship. It uses composite foreign keys experience_id and traffic_id to connect experiences with traffic types.

Traffic:
- Primary Key (PK): traffic_id
- Description: The Traffic entity represents different types of traffic conditions. Each traffic type is uniquely identified by traffic_id.

Experience_Weather:
- Foreign Keys (FK): experience_id, weather_id
- Description: The Experience_Weather entity is a junction table that links Experience and Weather entities in a many-to-many relationship. It uses composite foreign keys experience_id and weather_id to connect experiences with weather types.

Weather:

- Primary Key (PK): weather_id
- Description: The Weather entity represents different weather conditions. Each weather type is uniquely identified by weather_id.

Experience_Road:
- Foreign Keys (FK): experience_id, road_id
- Description: The Experience_Road entity is a junction table that links Experience and Road entities in a many-to-many relationship. It uses composite foreign keys experience_id and road_id to connect experiences with road types.

Road:
- Primary Key (PK): road_id
- Description: The Road entity represents different types of roads. Each road type is uniquely identified by road_id.

Experience_Maneuver:
- Foreign Keys (FK): experience_id, maneuver_id
- Description: The Experience_Maneuver entity is a junction table that links Experience and Maneuver entities in a many-to-many relationship. It uses composite foreign keys experience_id and maneuver_id to connect experiences with maneuver types.

Maneuver:
- Primary Key (PK): maneuver_id
- Description: The Maneuver entity represents different types of maneuvers. Each maneuver type is uniquely identified by maneuver_id.

Navigation:
- Primary Key (PK): navigation_id
- Description: The Navigation entity represents different navigation types. Each navigation type is uniquely identified by navigation_id.

Primary Keys (PK) are unique identifiers for each entity, ensuring that each record within an entity can be uniquely identified.
Foreign Keys (FK) establish relationships between entities, ensuring referential integrity.

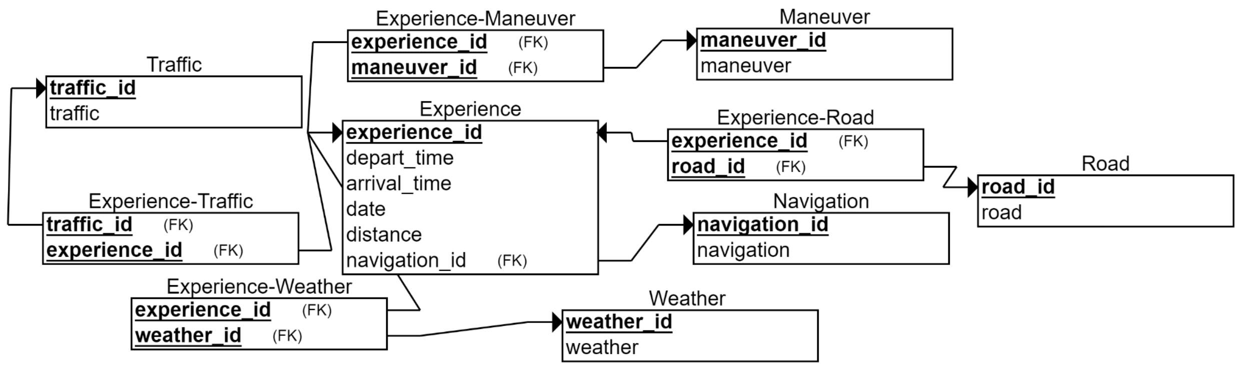
```
[Experience,PK:experience_id](110<)[Experience_Traffic,FK:experience_id,FK:traffic_id]
[Experience_Traffic,FK:experience_id,FK:traffic_id](>111)[Traffic,PK:traffic_id]

[Experience,PK:experience_id](110<)[Experience_Weather,FK:experience_id,FK:weather_id]
[Experience_Weather,FK:experience_id,FK:weather_id](>111)[Weather,PK:weather_id]
```

[Experience,PK:experience_id](110<)[Experience_Road,FK:experience_id,FK:doad_id]
[Experience_Road,FK:experience_id,FK:doad_id](>111)[Road,PK:doad_id]

[Experience,PK:experience_id](110<)[Experience_Maneuver,FK:experience_id,FK:maneuver_id
]
[Experience_Maneuver,FK:experience_id,FK:maneuver_id](>101)[Maneuver,PK:maneuver_id]

[Experience,PK:experience_id,FK:navigation_id](>011)[Navigation,PK:navigation_id]

Relational Schema



Experience (PK:experience_id, date, departure_time, arrival_time, distance, FK:navigation_id)

Experience_Traffic (FK:experience_id, FK:traffic_id)

Traffic (PK:traffic_id, traffic)

Experience_Weather (FK:experience_id, FK:weather_id)

Weather (PK:weather_id, weather)

Experience_Road (FK:experience_id, FK:road_id)

Road (PK:road_id, road)

Experience_Maneuver (FK:experience_id, FK:maneuver_id)

Maneuver (PK:maneuver_id, maneuver)

Navigation (PK:navigation_id, navigation)

Schema:

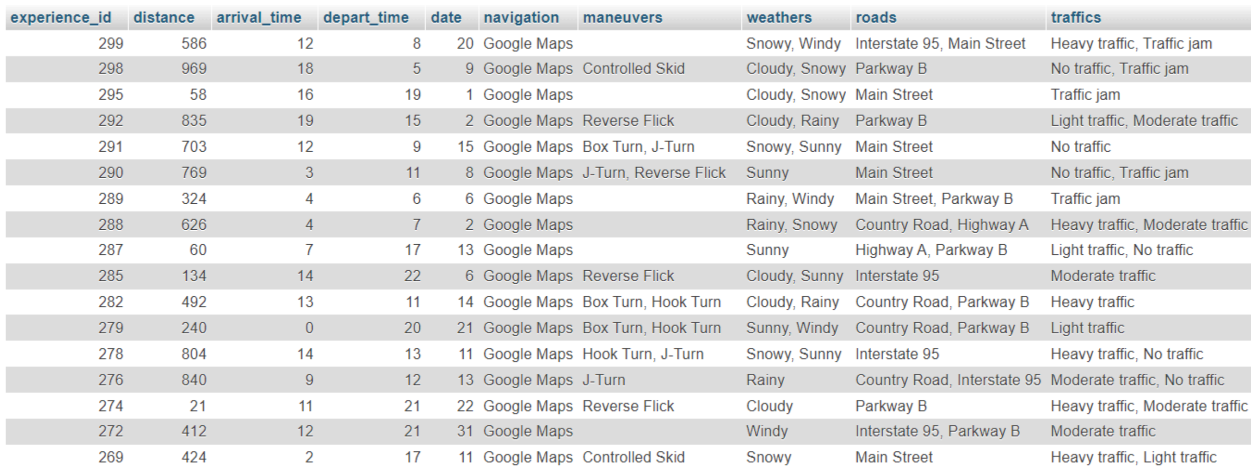
```
CREATE TABLE Weather
(
  weather_id INT NOT NULL,
  weather VARCHAR(20) NOT NULL,
  PRIMARY KEY (weather_id)
);

CREATE TABLE Road
(
  road_id INT NOT NULL,
  road VARCHAR(20) NOT NULL,
  PRIMARY KEY (road_id)
);

CREATE TABLE Navigation
(
  navigation_id INT NOT NULL,
  navigation VARCHAR(20) NOT NULL,
  PRIMARY KEY (navigation_id)
);

CREATE TABLE Maneuver
(
  maneuver VARCHAR(20) NOT NULL,
  maneuver_id INT NOT NULL,
  PRIMARY KEY (maneuver_id)
);

CREATE TABLE Traffic
(
  traffic_id INT NOT NULL,
  traffic VARCHAR(20) NOT NULL,
  PRIMARY KEY (traffic_id)
);

CREATE TABLE Experience
(
  depart_time INT NOT NULL,
  arrival_time INT NOT NULL,
  date INT NOT NULL,
  experience_id INT NOT NULL,
  distance INT NOT NULL,
  navigation_id INT NOT NULL,
  PRIMARY KEY (experience_id),
  FOREIGN KEY (navigation_id) REFERENCES Navigation(navigation_id)
);

CREATE TABLE Experience-Weather
(
  experience_id INT NOT NULL,
  weather_id INT NOT NULL,
  PRIMARY KEY (experience_id, weather_id),
  FOREIGN KEY (experience_id) REFERENCES Experience(experience_id),
  FOREIGN KEY (weather_id) REFERENCES Weather_(weather_id)
);

CREATE TABLE Experience-Road
(
  experience_id INT NOT NULL,
  road_id INT NOT NULL,
  PRIMARY KEY (experience_id, road_id),
  FOREIGN KEY (experience_id) REFERENCES Experience(experience_id),
  FOREIGN KEY (road_id) REFERENCES Road(road_id)
);

CREATE TABLE Experience-Maneuver
(
  experience_id INT NOT NULL,
  maneuver_id INT NOT NULL,
  PRIMARY KEY (experience_id, maneuver_id),
  FOREIGN KEY (experience_id) REFERENCES Experience(experience_id),
  FOREIGN KEY (maneuver_id) REFERENCES Maneuver(maneuver_id)
);

CREATE TABLE Experience-Traffic
(
  traffic_id INT NOT NULL,
  experience_id INT NOT NULL,
  PRIMARY KEY (traffic_id, experience_id),
  FOREIGN KEY (traffic_id) REFERENCES Traffic(traffic_id),
  FOREIGN KEY (experience_id) REFERENCES Experience(experience_id)
);
```

Queries

This SQL query creates a table that connects all tables and displays data in one row.

```
SELECT
  e.experience_id,
  e.distance,
  e.arrival_time,
  e.depart_time,
  e.date,
  n.navigation,
  CONCAT_WS(', ', (
    SELECT GROUP_CONCAT(m.maneuver ORDER BY m.maneuver SEPARATOR ', ')
    FROM Experience_Maneuver em
    JOIN Maneuver m ON em.maneuver_id = m.maneuver_id
    WHERE em.experience_id = e.experience_id
  )) AS maneuvers,
  CONCAT_WS(', ', (
    SELECT GROUP_CONCAT(w.weather ORDER BY w.weather SEPARATOR ', ')
    FROM Experience_Weather ew
    JOIN Weather w ON ew.weather_id = w.weather_id
    WHERE ew.experience_id = e.experience_id
  )) AS weathers,
  CONCAT_WS(', ', (
    SELECT GROUP_CONCAT(r.road ORDER BY r.road SEPARATOR ', ')
    FROM Experience_Road er
    JOIN Road r ON er.road_id = r.road_id
    WHERE er.experience_id = e.experience_id
  )) AS roads,
  CONCAT_WS(', ', (
    SELECT GROUP_CONCAT(t.traffic ORDER BY t.traffic SEPARATOR ', ')
    FROM Experience_Traffic et
    JOIN Traffic t ON et.traffic_id = t.traffic_id
    WHERE et.experience_id = e.experience_id
  )) AS traffics
FROM Experience e
JOIN Navigation n ON e.navigation_id = n.navigation_id
GROUP BY e.experience_id;
```

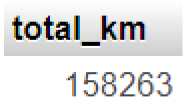
The result:



experience_id	distance	arrival_time	depart_time	date	navigation	maneuvers	weathers	roads	traffics
299	586	12	8	20	Google Maps		Snowy, Windy	Interstate 95, Main Street	Heavy traffic, Traffic jam
298	969	18	5	9	Google Maps	Controlled Skid	Cloudy, Snowy	Parkway B	No traffic, Traffic jam
295	58	16	19	1	Google Maps		Cloudy, Snowy	Main Street	Traffic jam
292	835	19	15	2	Google Maps	Reverse Flick	Cloudy, Rainy	Parkway B	Light traffic, Moderate traffic
291	703	12	9	15	Google Maps	Box Turn, J-Turn	Snowy, Sunny	Main Street	No traffic
290	769	3	11	8	Google Maps	J-Turn, Reverse Flick	Sunny	Main Street	No traffic, Traffic jam
289	324	4	6	6	Google Maps		Rainy, Windy	Main Street, Parkway B	Traffic jam
288	626	4	7	2	Google Maps		Rainy, Snowy	Country Road, Highway A	Heavy traffic, Moderate traffic
287	60	7	17	13	Google Maps		Sunny	Highway A, Parkway B	Light traffic, No traffic
285	134	14	22	6	Google Maps	Reverse Flick	Cloudy, Sunny	Interstate 95	Moderate traffic
282	492	13	11	14	Google Maps	Box Turn, Hook Turn	Cloudy, Rainy	Country Road, Parkway B	Heavy traffic
279	240	0	20	21	Google Maps	Box Turn, Hook Turn	Sunny, Windy	Country Road, Parkway B	Light traffic
278	804	14	13	11	Google Maps	Hook Turn, J-Turn	Snowy, Sunny	Interstate 95	Heavy traffic, No traffic
276	840	9	12	13	Google Maps	J-Turn	Rainy	Country Road, Interstate 95	Moderate traffic, No traffic
274	21	11	21	22	Google Maps	Reverse Flick	Cloudy	Parkway B	Heavy traffic, Moderate traffic
272	412	12	21	31	Google Maps		Windy	Interstate 95, Parkway B	Moderate traffic
269	424	2	17	11	Google Maps	Controlled Skid	Snowy	Main Street	Heavy traffic, Light traffic

This query shows the total passed kilometers from all entries:

```
SELECT SUM(e.distance) AS total_km
FROM experience AS e
```



total_km
158263

This query shows how many maneuvers have been performed in total:

```
SELECT m.maneuver, COUNT(em.experience_id) AS total
FROM maneuver m
LEFT JOIN experience_maneuver em ON em.maneuver_id = m.maneuver_id
GROUP BY m.maneuver_id
```

maneuver	total
J-Turn	62
Hook Turn	61
Box Turn	71
Controlled Skid	56
Reverse Flick	72

This query shows how many kilometers were passed on each road:

```
SELECT r.road, SUM(e.distance) AS total_km
FROM experience_road AS er
JOIN road r ON er.road_id = r.road_id
JOIN experience e ON er.experience_id = e.experience_id
GROUP BY r.road_id
```

road	total_km
Highway A	45522
Main Street	54181
Country Road	47431
Interstate 95	54010
Parkway B	39499

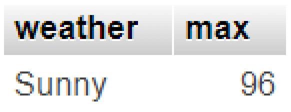
This query shows how many drives has been performed during day time:

```
SELECT COUNT(e.experience_id) AS day_time
FROM experience AS e
WHERE e.depart_time > 6 AND e.depart_time < 18
AND e.arrival_time > 6 AND e.arrival_time < 18
AND e.depart_time < e.arrival_time
```

day_time
37

This query shows the most frequent weather type:

```sql
SELECT sub.weather, MAX(sub.occurences) AS max
FROM (
    SELECT w.weather, COUNT(*) AS occurences
    FROM experience AS e
    JOIN experience_weather ew ON e.experience_id = ew.experience_id
    JOIN weather w ON w.weather_id = ew.weather_id
    GROUP BY w.weather
) AS sub
```



weather	max
Sunny	96

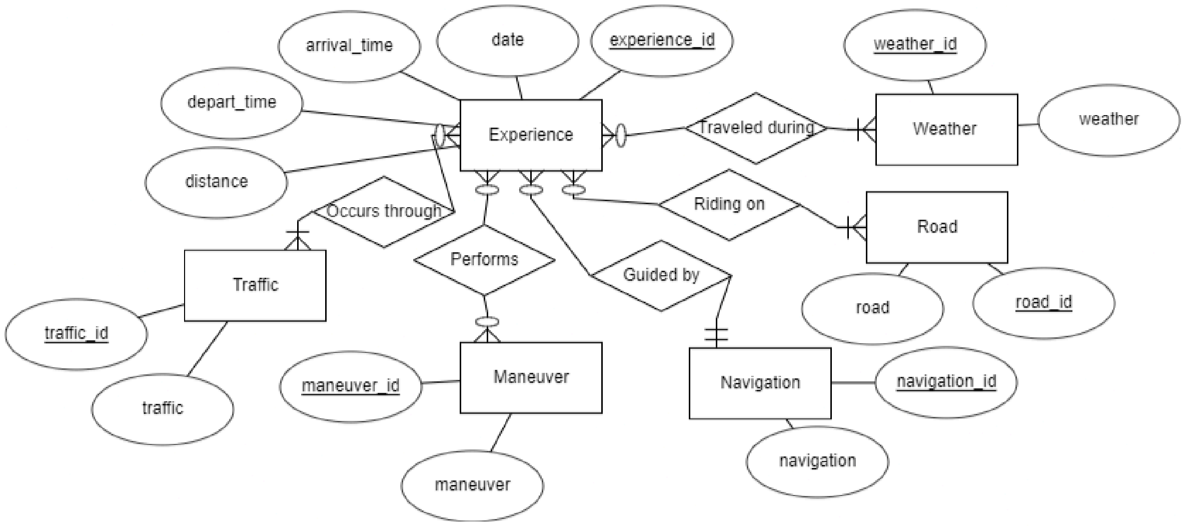
Part 2: MongoDB Database

1) The database structure imported into MongoDB Atlas from JSON data is considered normalized because it adheres to principles that minimize redundancy and dependency. It eliminates redundancy by storing data in separate collections/entities and establishes relationships through references.

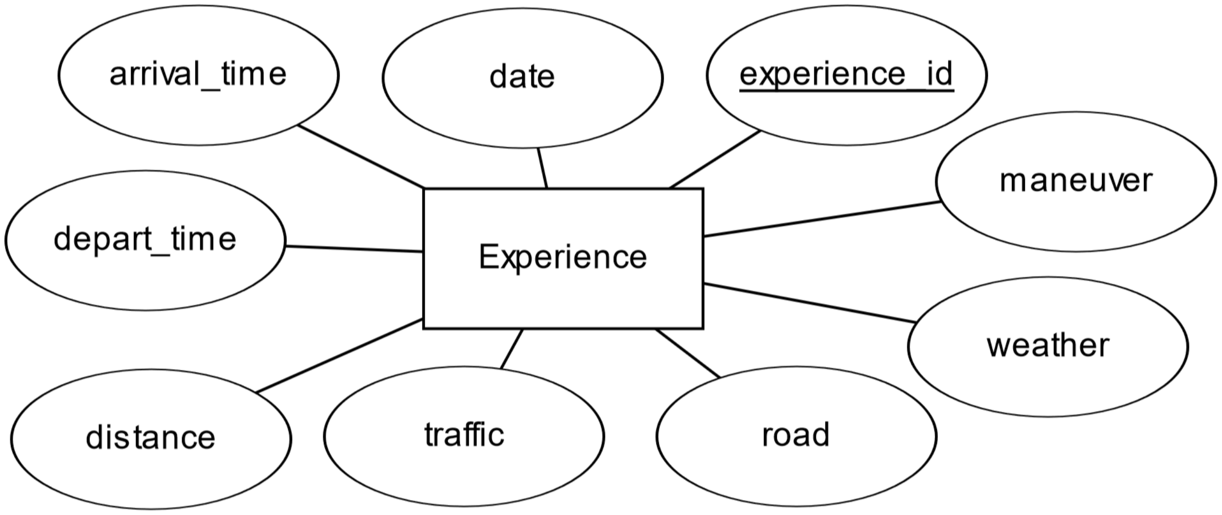
2) Denormalization in databases involves consolidating data into fewer tables or documents compared to the normalized approach. It enhances read performance by reducing the need for complex joins and simplifies querying. Denormalization also reduces complexity, making database management easier. It improves query performance and is particularly beneficial for read-heavy workloads.

Denormalization:

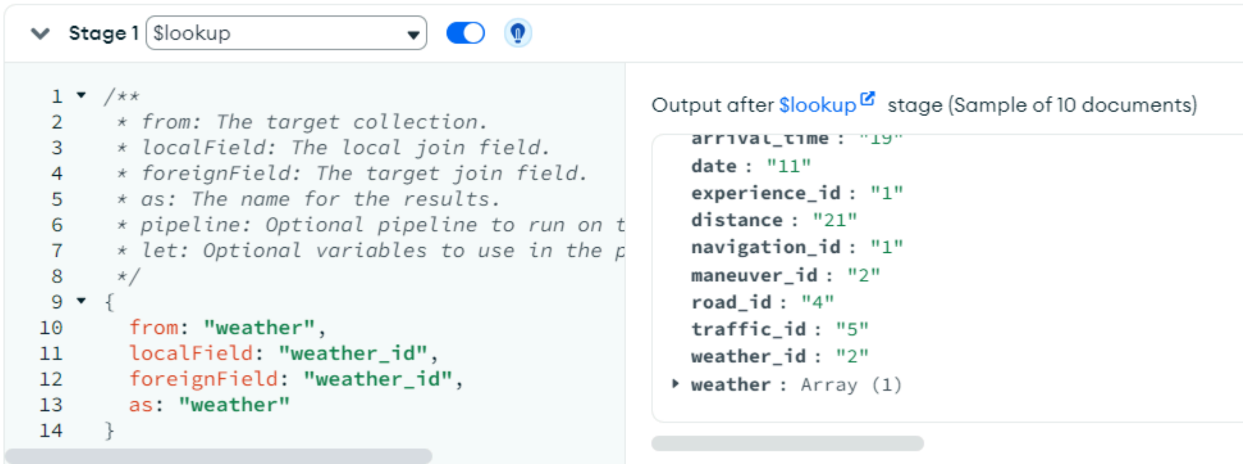
Before:



After:



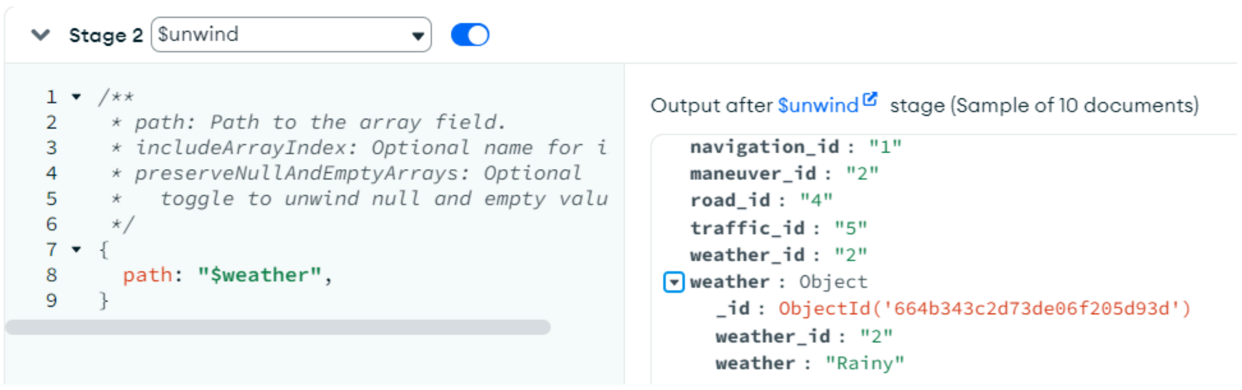
The lookup stage joins the selected table to the document based on the selected keyword. In the picture, it adds the weather document to the experience document as an additional field.



```
/**
 * from: The target collection.
 * localField: The local join field.
 * foreignField: The target join field.
 * as: The name for the results.
 * pipeline: Optional pipeline to run on t
 * let: Optional variables to use in the p
 */
{
  from: "weather",
  localField: "weather_id",
  foreignField: "weather_id",
  as: "weather"
}
```

```
arrival_time : "19"
date : "11"
experience_id : "1"
distance : "21"
navigation_id : "1"
maneuver_id : "2"
road_id : "4"
traffic_id : "5"
weather_id : "2"
▸ weather : Array (1)
```

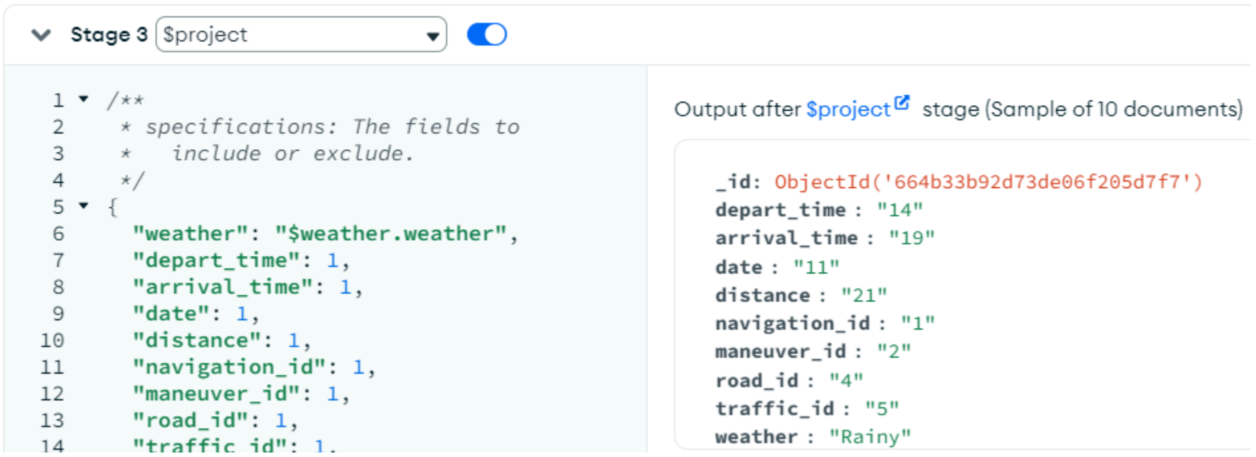
The lookup adds the documents as an array. The unwind keyword transforms the array into an object.



```
/**
 * path: Path to the array field.
 * includeArrayIndex: Optional name for i
 * preserveNullAndEmptyArrays: Optional
 *   toggle to unwind null and empty valu
 */
{
  path: "$weather",
}
```

```
navigation_id : "1"
maneuver_id : "2"
road_id : "4"
traffic_id : "5"
weather_id : "2"
weather : Object
  _id : ObjectId('664b343c2d73de06f205d93d')
  weather_id : "2"
  weather : "Rainy"
```

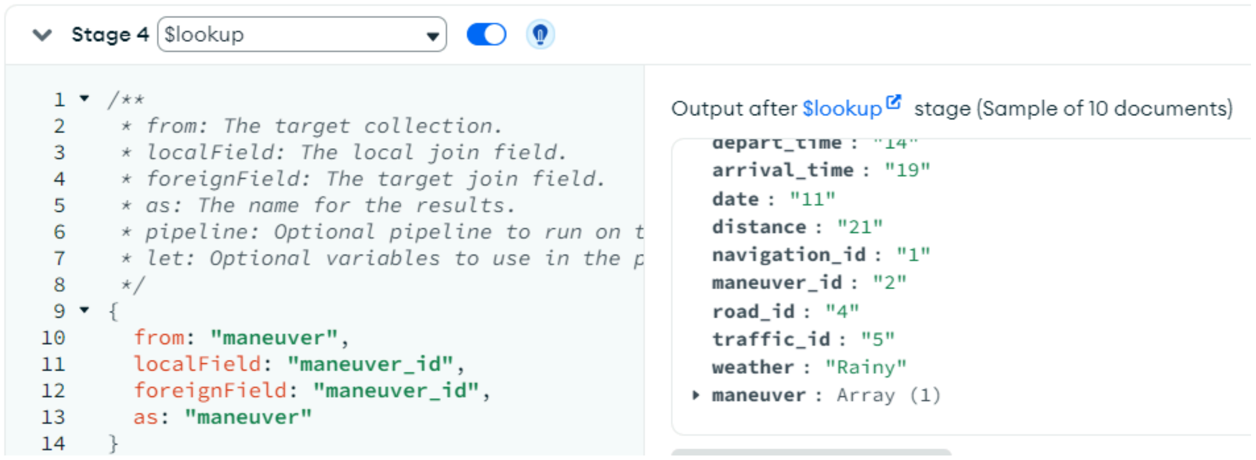
The project keyword removes the weather_id, and sets the value of weather field to subfield “weather”.



```
/**
 * specifications: The fields to
 *   include or exclude.
 */
{
  "weather": "$weather.weather",
  "depart_time": 1,
  "arrival_time": 1,
  "date": 1,
  "distance": 1,
  "navigation_id": 1,
  "maneuver_id": 1,
  "road_id": 1,
  "traffic_id": 1,
```

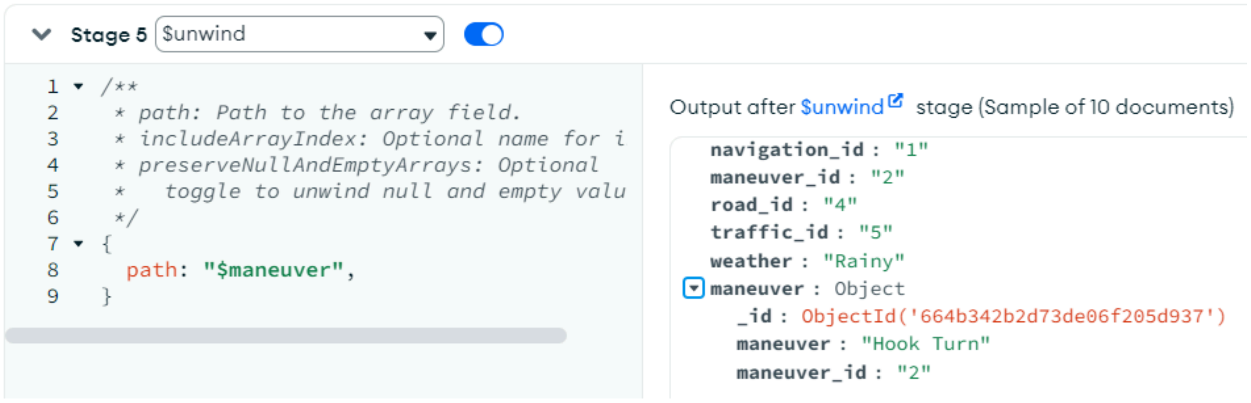
```
_id: ObjectId('664b33b92d73de06f205d7f7')
depart_time : "14"
arrival_time : "19"
date : "11"
distance : "21"
navigation_id : "1"
maneuver_id : "2"
road_id : "4"
traffic_id : "5"
weather : "Rainy"
```

The cycle is repeated for each one-to-many relationship with the only difference being the content of the project field.



```
/**
 * from: The target collection.
 * localField: The local join field.
 * foreignField: The target join field.
 * as: The name for the results.
 * pipeline: Optional pipeline to run on t
 * let: Optional variables to use in the p
 */
{
  from: "maneuver",
  localField: "maneuver_id",
  foreignField: "maneuver_id",
  as: "maneuver"
}
```

```
depart_time : "14"
arrival_time : "19"
date : "11"
distance : "21"
navigation_id : "1"
maneuver_id : "2"
road_id : "4"
traffic_id : "5"
weather : "Rainy"
maneuver : Array (1)
```

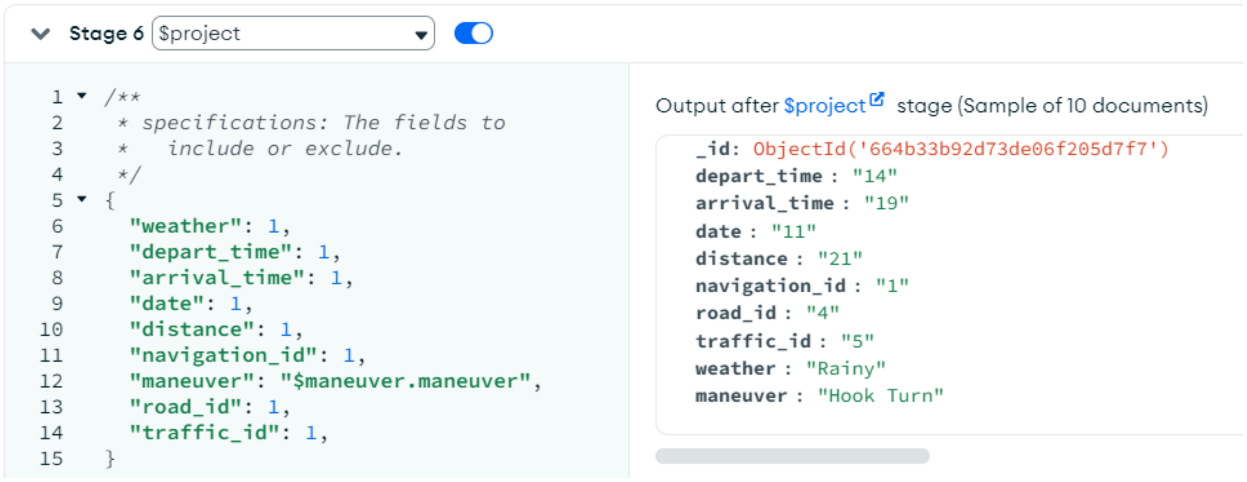


Stage 5 $unwind

```
/**
  * path: Path to the array field.
  * includeArrayIndex: Optional name for i
  * preserveNullAndEmptyArrays: Optional
  *   toggle to unwind null and empty valu
  */
{
  path: "$maneuver",
}
```

Output after $unwind stage (Sample of 10 documents)

```
navigation_id : "1"
maneuver_id : "2"
road_id : "4"
traffic_id : "5"
weather : "Rainy"
maneuver : Object
  _id : ObjectId('664b342b2d73de06f205d937')
  maneuver : "Hook Turn"
  maneuver_id : "2"
```

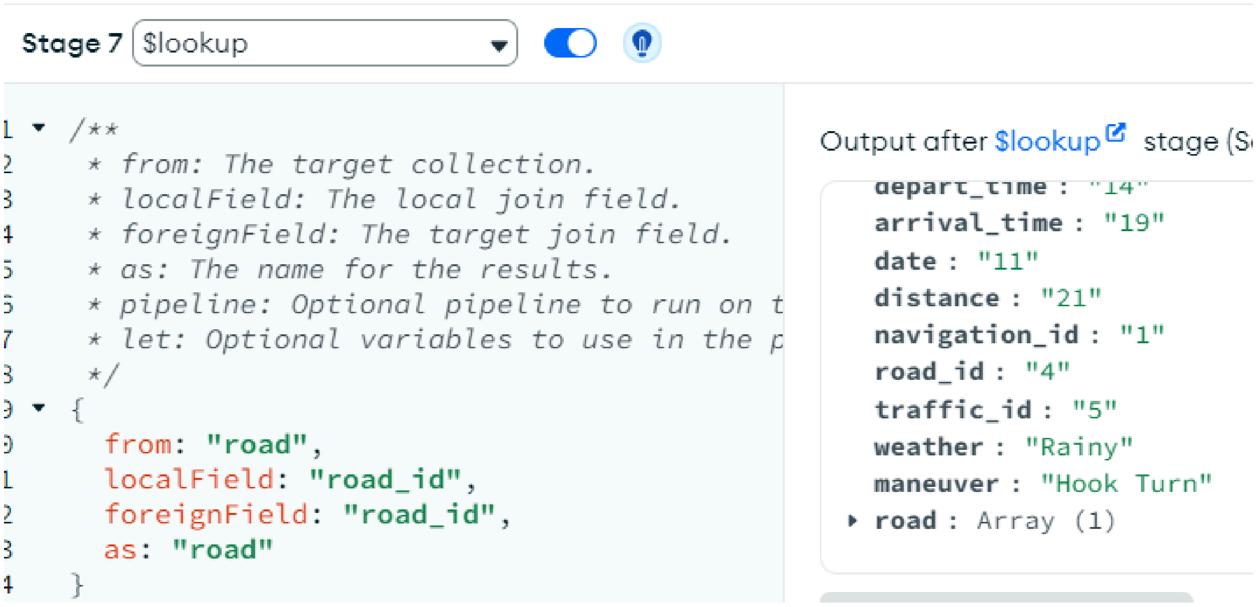


Stage 6 $project

```
/**
 * specifications: The fields to
 *   include or exclude.
 */
{
  "weather": 1,
  "depart_time": 1,
  "arrival_time": 1,
  "date": 1,
  "distance": 1,
  "navigation_id": 1,
  "maneuver": "$maneuver.maneuver",
  "road_id": 1,
  "traffic_id": 1,
}
```

Output after $project stage (Sample of 10 documents)

```
_id: ObjectId('664b33b92d73de06f205d7f7')
depart_time : "14"
arrival_time : "19"
date : "11"
distance : "21"
navigation_id : "1"
road_id : "4"
traffic_id : "5"
weather : "Rainy"
maneuver : "Hook Turn"
```

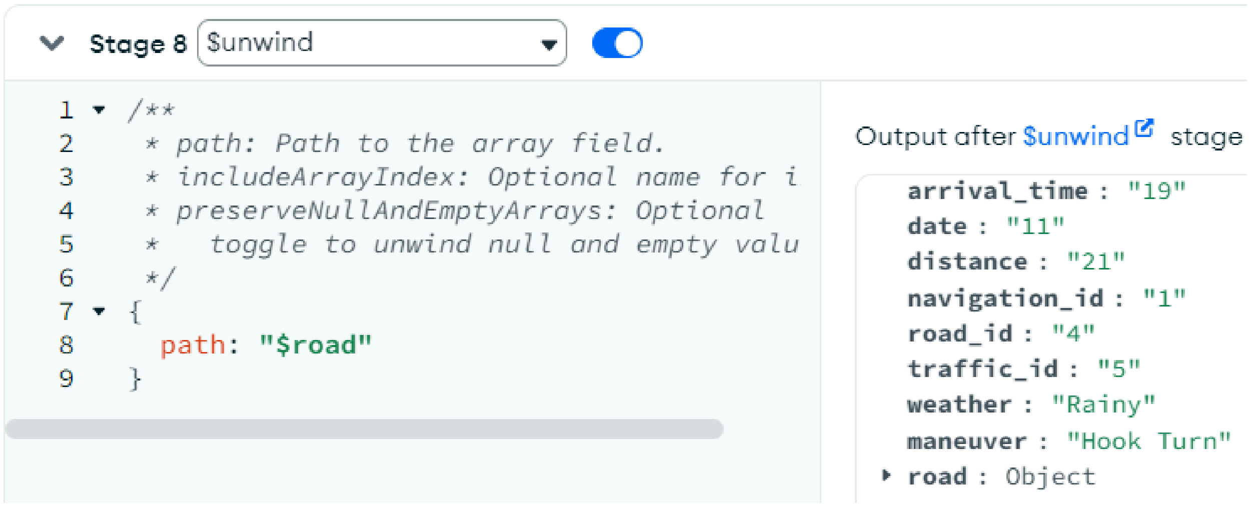


Stage 7 $lookup

```
/**
  * from: The target collection.
  * localField: The local join field.
  * foreignField: The target join field.
  * as: The name for the results.
  * pipeline: Optional pipeline to run on t
  * let: Optional variables to use in the p
  */
{
  from: "road",
  localField: "road_id",
  foreignField: "road_id",
  as: "road"
}
```

Output after $lookup stage (S

```
depart_time : "14"
arrival_time : "19"
date : "11"
distance : "21"
navigation_id : "1"
road_id : "4"
traffic_id : "5"
weather : "Rainy"
maneuver : "Hook Turn"
road : Array (1)
```

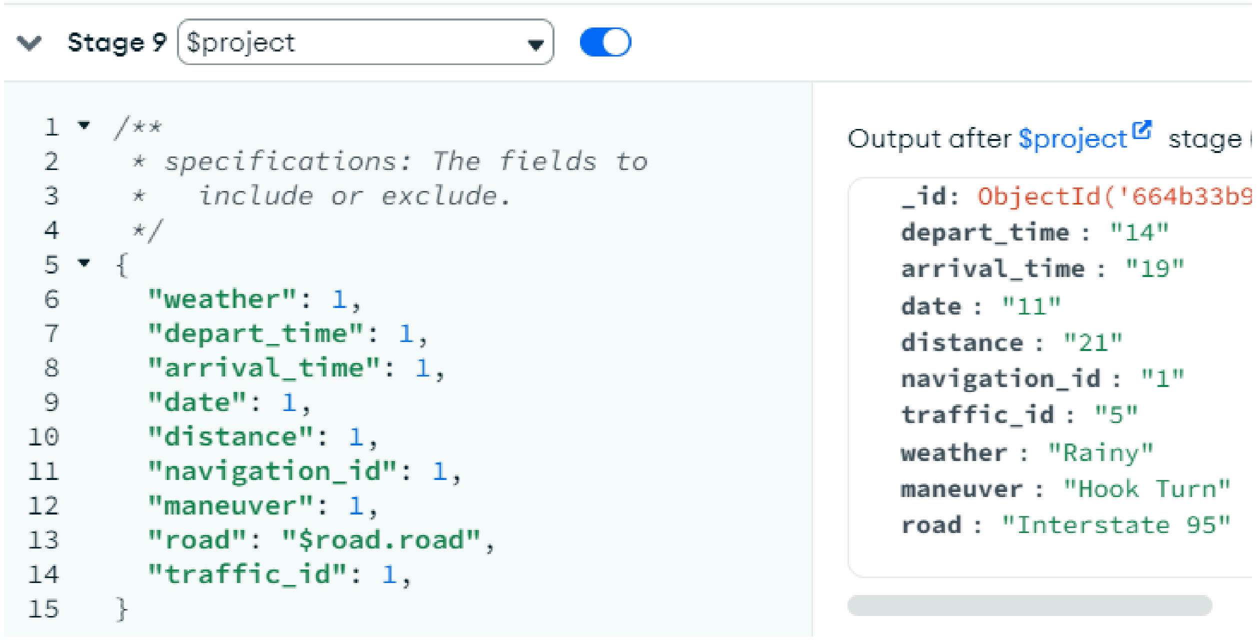


Stage 8 $unwind

```
/**
 * path: Path to the array field.
 * includeArrayIndex: Optional name for i
 * preserveNullAndEmptyArrays: Optional
 *   toggle to unwind null and empty valu
 */
{
  path: "$road"
}
```

Output after $unwind stage

```
arrival_time : "19"
date : "11"
distance : "21"
navigation_id : "1"
road_id : "4"
traffic_id : "5"
weather : "Rainy"
maneuver : "Hook Turn"
road : Object
```

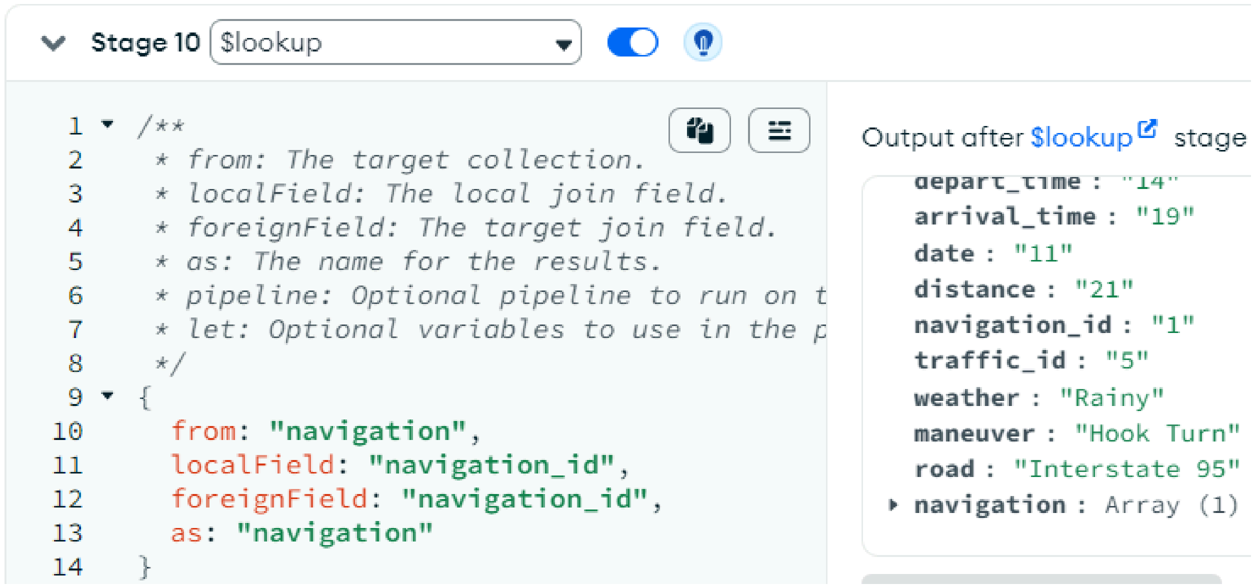


Stage 9 $project

```
/**
 * specifications: The fields to
 *   include or exclude.
 */
{
  "weather": 1,
  "depart_time": 1,
  "arrival_time": 1,
  "date": 1,
  "distance": 1,
  "navigation_id": 1,
  "maneuver": 1,
  "road": "$road.road",
  "traffic_id": 1,
}
```

Output after $project stage

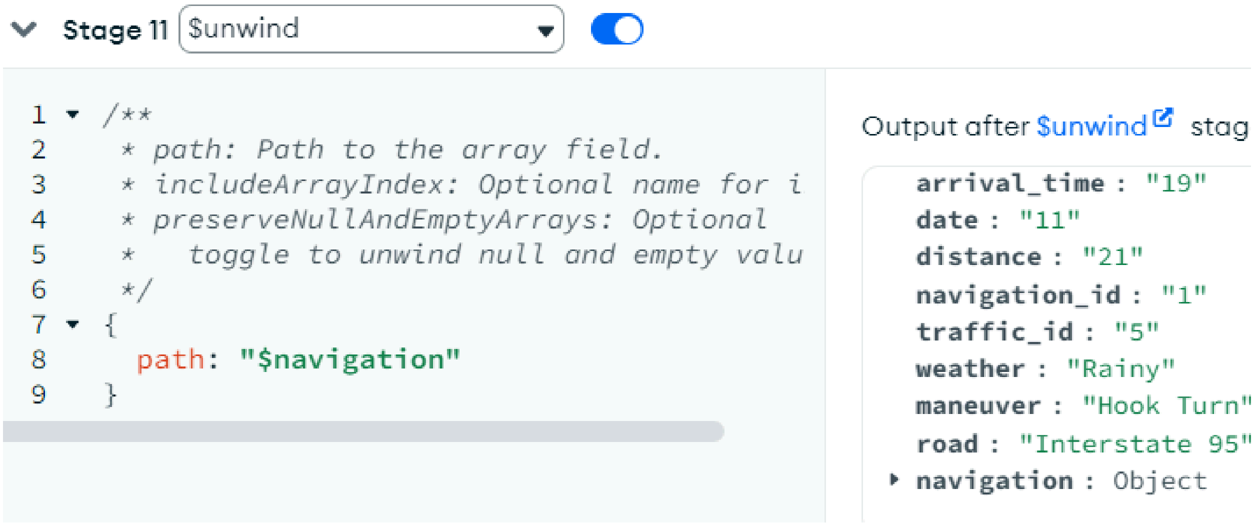
```
_id: ObjectId('664b33b9
depart_time : "14"
arrival_time : "19"
date : "11"
distance : "21"
navigation_id : "1"
traffic_id : "5"
weather : "Rainy"
maneuver : "Hook Turn"
road : "Interstate 95"
```


Stage 10 $lookup

```
/**
 * from: The target collection.
 * localField: The local join field.
 * foreignField: The target join field.
 * as: The name for the results.
 * pipeline: Optional pipeline to run on t
 * let: Optional variables to use in the p
 */
{
  from: "navigation",
  localField: "navigation_id",
  foreignField: "navigation_id",
  as: "navigation"
}
```

Output after $lookup stage

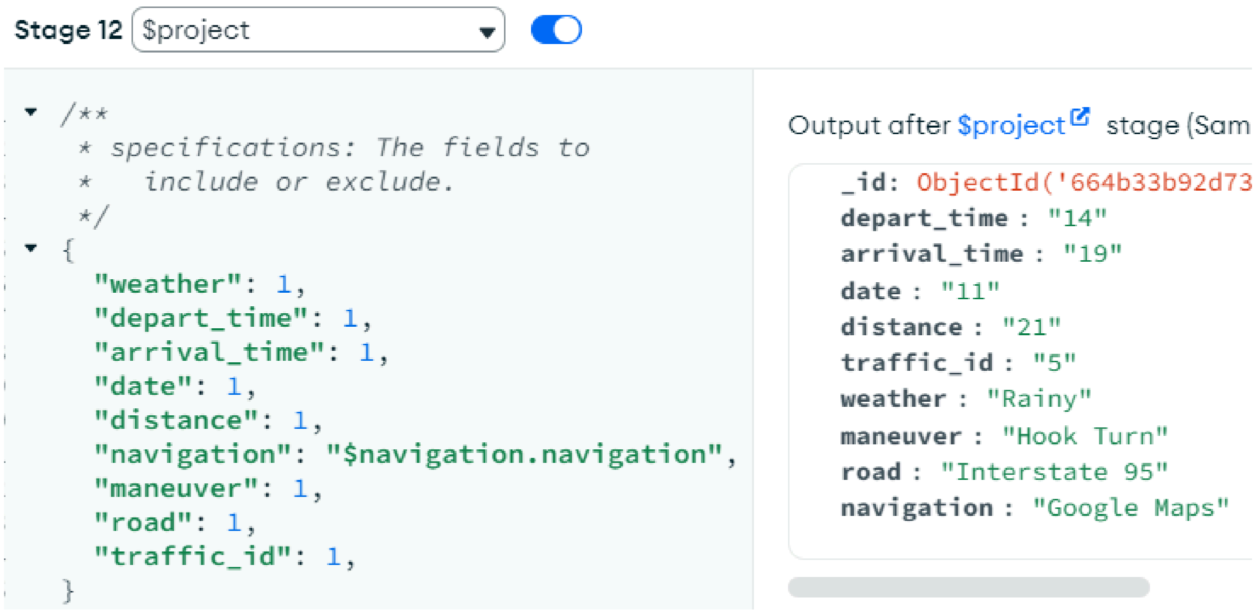
```
depart_time : "14"
arrival_time : "19"
date : "11"
distance : "21"
navigation_id : "1"
traffic_id : "5"
weather : "Rainy"
maneuver : "Hook Turn"
road : "Interstate 95"
▸ navigation : Array (1)
```


Stage 11 $unwind

```
/**
 * path: Path to the array field.
 * includeArrayIndex: Optional name for i
 * preserveNullAndEmptyArrays: Optional
 *   toggle to unwind null and empty valu
 */
{
  path: "$navigation"
}
```

Output after $unwind stag

```
arrival_time : "19"
date : "11"
distance : "21"
navigation_id : "1"
traffic_id : "5"
weather : "Rainy"
maneuver : "Hook Turn"
road : "Interstate 95"
▸ navigation : Object
```

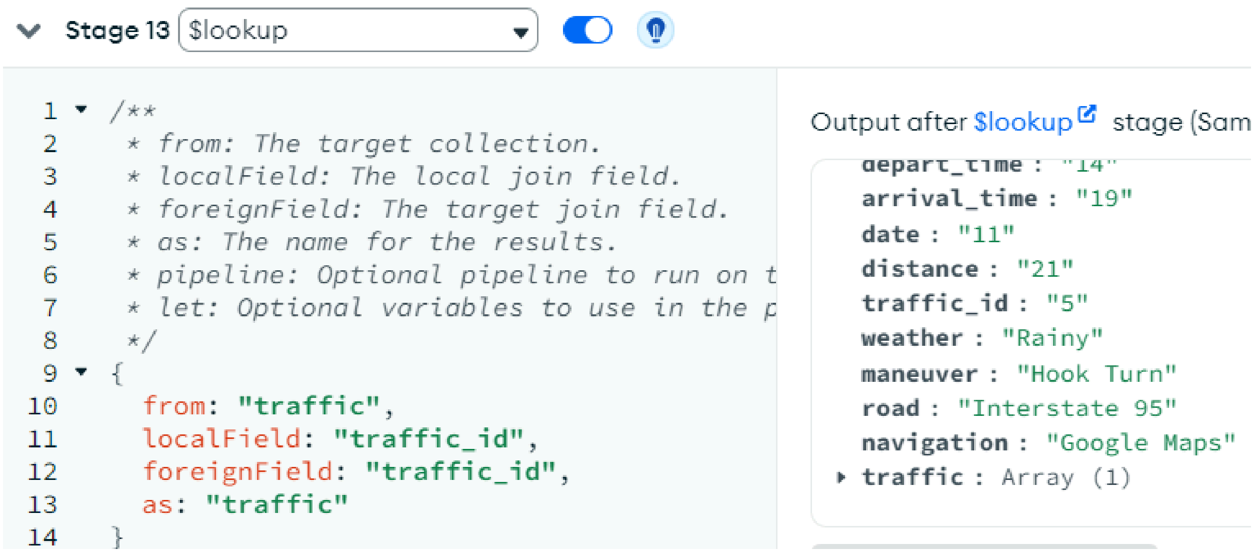


Stage 12 $project

```
/**
 * specifications: The fields to
 *   include or exclude.
 */
{
  "weather": 1,
  "depart_time": 1,
  "arrival_time": 1,
  "date": 1,
  "distance": 1,
  "navigation": "$navigation.navigation",
  "maneuver": 1,
  "road": 1,
  "traffic_id": 1,
}
```

Output after $project stage (Sam

```
_id: ObjectId('664b33b92d73
depart_time : "14"
arrival_time : "19"
date : "11"
distance : "21"
traffic_id : "5"
weather : "Rainy"
maneuver : "Hook Turn"
road : "Interstate 95"
navigation : "Google Maps"
```

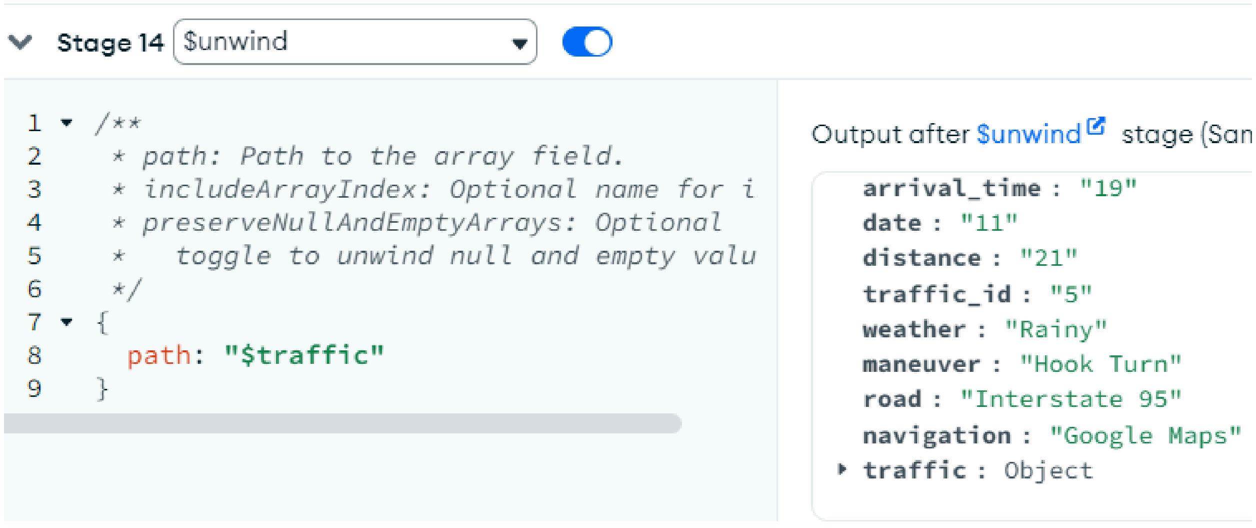


Stage 13 $lookup

```
/**
 * from: The target collection.
 * localField: The local join field.
 * foreignField: The target join field.
 * as: The name for the results.
 * pipeline: Optional pipeline to run on t
 * let: Optional variables to use in the p
 */
{
  from: "traffic",
  localField: "traffic_id",
  foreignField: "traffic_id",
  as: "traffic"
}
```

Output after $lookup stage (Sam

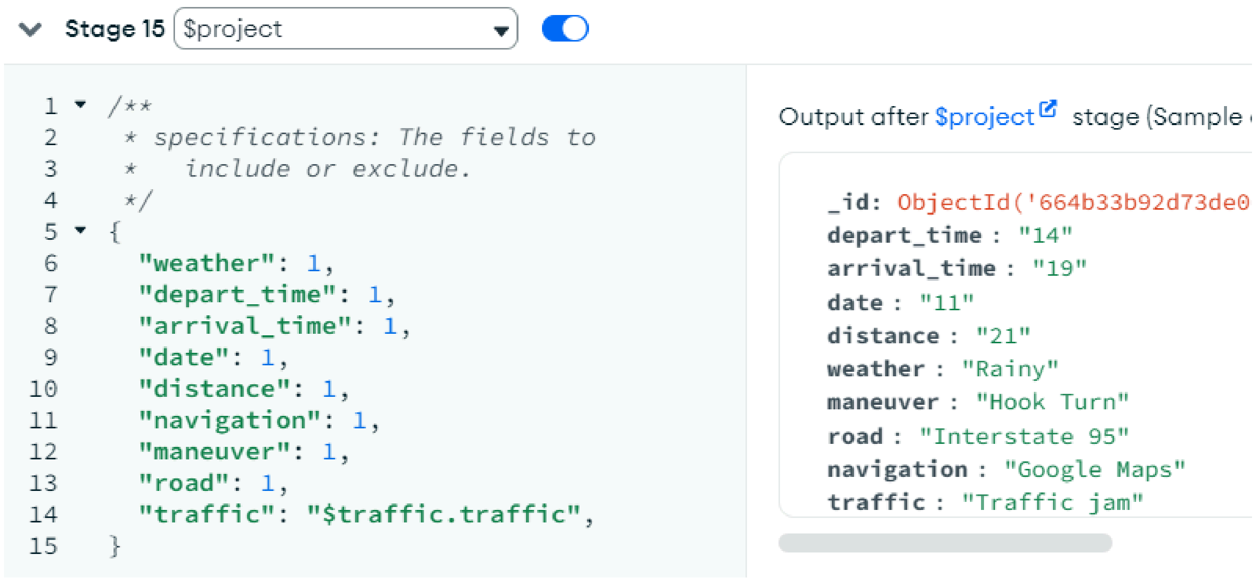
```
depart_time : "14"
arrival_time : "19"
date : "11"
distance : "21"
traffic_id : "5"
weather : "Rainy"
maneuver : "Hook Turn"
road : "Interstate 95"
navigation : "Google Maps"
▸ traffic : Array (1)
```



```
/**
 * path: Path to the array field.
 * includeArrayIndex: Optional name for i
 * preserveNullAndEmptyArrays: Optional
 *   toggle to unwind null and empty valu
 */
{
  path: "$traffic"
}
```

Output after $unwind stage (San

```
arrival_time : "19"
date : "11"
distance : "21"
traffic_id : "5"
weather : "Rainy"
maneuver : "Hook Turn"
road : "Interstate 95"
navigation : "Google Maps"
▸ traffic : Object
```



```
/**
 * specifications: The fields to
 *   include or exclude.
 */
{
  "weather": 1,
  "depart_time": 1,
  "arrival_time": 1,
  "date": 1,
  "distance": 1,
  "navigation": 1,
  "maneuver": 1,
  "road": 1,
  "traffic": "$traffic.traffic",
}
```

Output after $project stage (Sample

```
_id: ObjectId('664b33b92d73de0
depart_time : "14"
arrival_time : "19"
date : "11"
distance : "21"
weather : "Rainy"
maneuver : "Hook Turn"
road : "Interstate 95"
navigation : "Google Maps"
traffic : "Traffic jam"
```

Full aggregation code

```
[
 {
   $lookup:
     {
       from: "weather",
       localField: "weather_id",
       foreignField: "weather_id",
       as: "weather"
     }
 },
 {
  $unwind:
    {
      path: "$weather"
    }
},
{
  $project:
    {
      weather: "$weather.weather",
      depart_time: 1,
      arrival_time: 1,
      date: 1,
      distance: 1,
      navigation_id: 1,
      maneuver_id: 1,
      road_id: 1,
      traffic_id: 1
    }
},
{
  $lookup:
    {
      from: "maneuver",
      localField: "maneuver_id",
      foreignField: "maneuver_id",
      as: "maneuver"
    }
},
{
  $unwind:
    {
      path: "$maneuver"
    }
},
{
  $project:
    {
      weather: 1,
      depart_time: 1,
      arrival_time: 1,
      date: 1,
      distance: 1,
      navigation_id: 1,
      maneuver: "$maneuver.maneuver",
      road_id: 1,
      traffic_id: 1
    }
},
{
  $lookup:
    {
      from: "road",
      localField: "road_id",
      foreignField: "road_id",
      as: "road"
    }
},
{
  $unwind:
    {
      path: "$road"
    }
},
{
  $project:
    {
      weather: 1,
      depart_time: 1,
      arrival_time: 1,
      date: 1,
      distance: 1,
      navigation_id: 1,
      maneuver: 1,
      road: "$road.road",
      traffic_id: 1
    }
},
{
  $lookup:
    {
      from: "navigation",
      localField: "navigation_id",
      foreignField: "navigation_id",
      as: "navigation"
    }
  },
  {
   $unwind:
    {
      path: "$navigation"
    }
  },
  {
   $project:
    {
      weather: 1,
      depart_time: 1,
      arrival_time: 1,
      date: 1,
      distance: 1,
      navigation: "$navigation.navigation",
      maneuver: 1,
      road: 1,
      traffic_id: 1
    }
  },
  {
   $lookup:
    {
      from: "traffic",
      localField: "traffic_id",
      foreignField: "traffic_id",
      as: "traffic"
    }
  },
  {
   $unwind:
    {
      path: "$traffic"
    }
  },
  {
    $project:
      {
        weather: 1,
        depart_time: 1,
        arrival_time: 1,
        date: 1,
        distance: 1,
        navigation: 1,
        maneuver: 1,
        road: 1,
        traffic: "$traffic.traffic"
      }
  }
]
```