AutoAvenue

CIS 5500 Final Project

Hassan Rizwan
Peter Proneca
William Qi
Vikramaditya Singh

Problem & Solution

The used car market is not transparent and new buyers do not have the ability to make contextualized data-driven decisions when buying from their local buyers

- **Optimize your search** for cars based on year of production, mileage, mpg, and other factors and compare against other cars
- See **reviews** from other users

Discovery within the used car market is fairly limited and automated recommendations are often price gouging

- Through our **Hidden Gems** feature, we present cars that our being sold at a significant discount to the market average but yet are in great condition and have good reviews

Market wide data is often not considered during decision making in secondary sales

- View **safety** and owner history, **accident** trends per car

Preview

AutoAvenue   SEARCH   REVIEWS   GEMS   SAFETY RANKING

Check out our featured car of the day:

MERCEDES-BENZ E-CLASS, 2016

Top Cars By Rating

Make	Model	Rating
AUDI	S7	4.8125
BMW	M2	4.8
ACURA	NSX	4.6912
KIA	STINGER	4.6214
PORSCHE	MACAN	4.605

Rows per page: 5 ▾   1–5 of more than 5   ‹ ›

Datasets

Used Cars

- CSV Format
- 593,378 Rows
- Contains information about manufacturer, model, year, mileage, drivetrain, fuel type, mpg, and price, among other specific characteristics

DATA WRANGLING

- Drop NAs and duplicates
- Drop auxillary columns such as paint color, interior color, seller name, and seller rating
- Cast mileage, year, and price as integer

Reviews

- CSV Format
- 298,567 Rows
- Contains review information such as rating and comments

- Drop NAs and duplicates
- Drop auxillary columns such as paint color, interior color, seller name, and seller rating
- Cast mileage, year, and price as integer

Data Pre-Processing
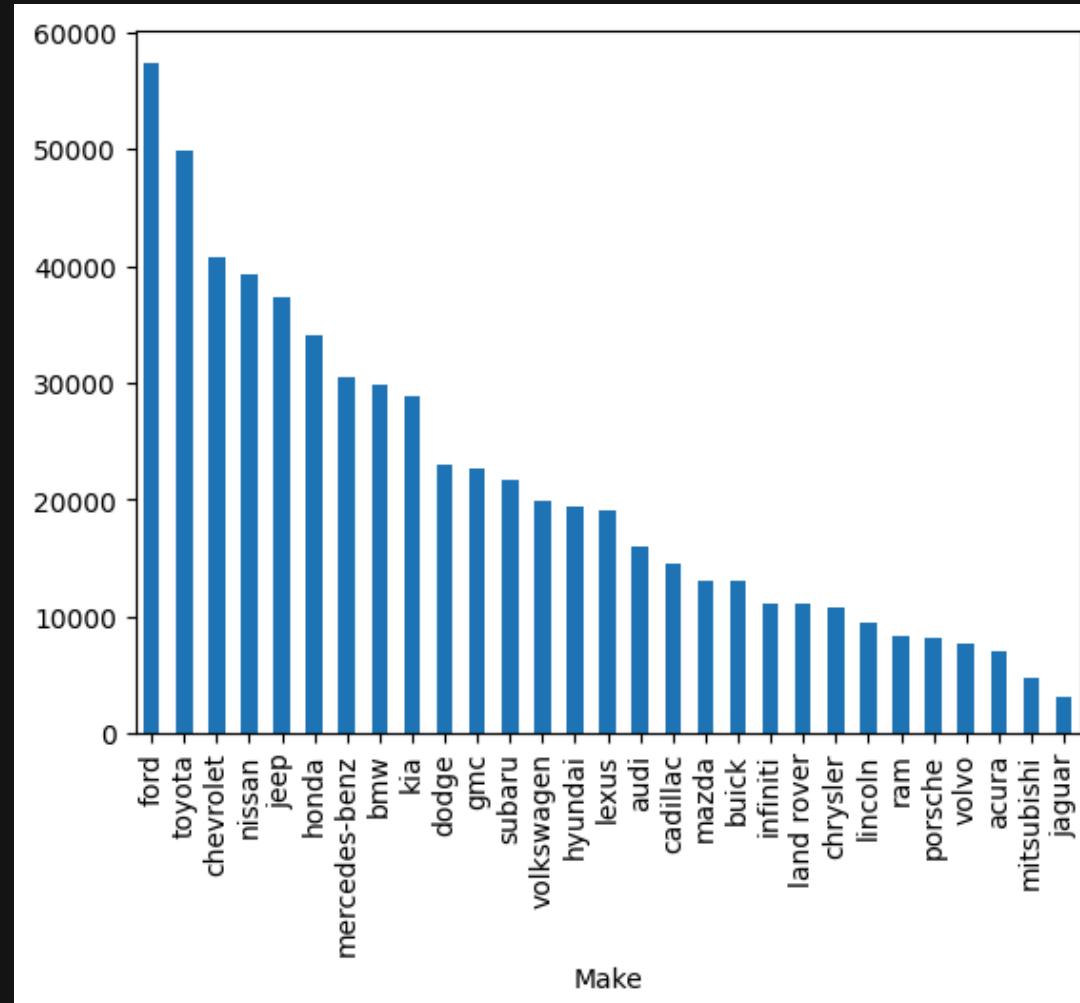
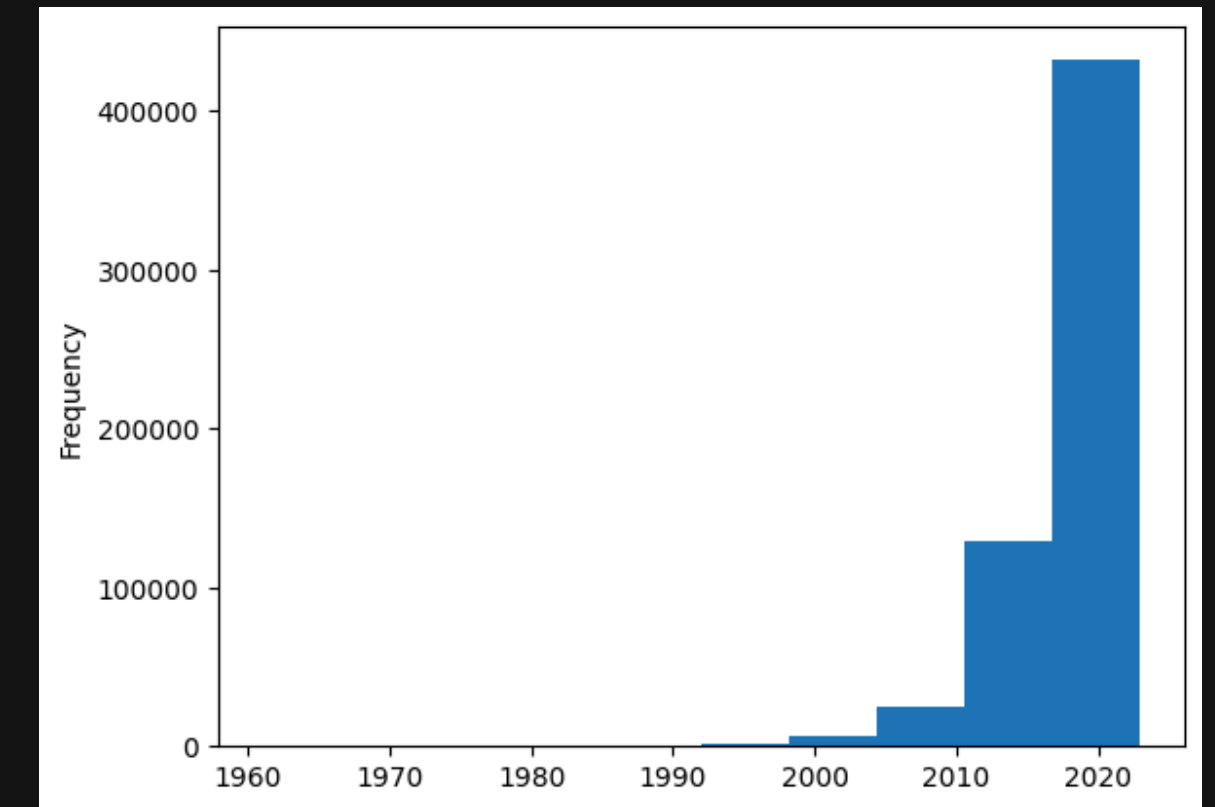Distribution of companies in UsedCars

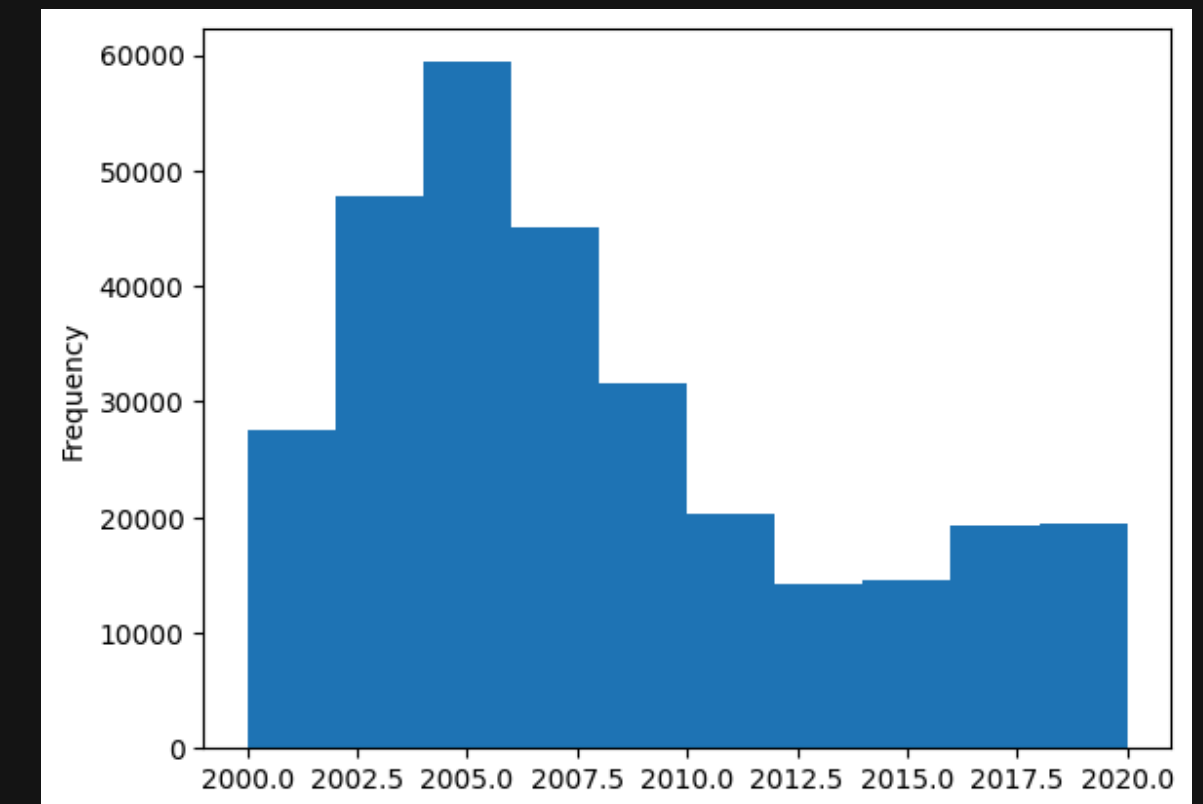Distribution of years in Used Cars

```python
1 drivetrain_mapping = {
2     'Front-wheel Drive': 'FWD',
3     'Front Wheel Drive': 'FWD',
4     'FWD': 'FWD',
5     'All-wheel Drive': 'AWD',
6     'All Wheel Drive': 'AWD',
7     'AWD': 'AWD',
8     'Four-wheel Drive': '4WD',
9     'Four Wheel Drive': '4WD',
10    '4WD': '4WD',
11    '4X4': '4WD',
12    'Rear-wheel Drive': 'RWD',
13    'RWD': 'RWD'
14 }
15 uc_df['Drivetrain'] = uc_df['Drivetrain'].map(drivetrain_mapping)
```
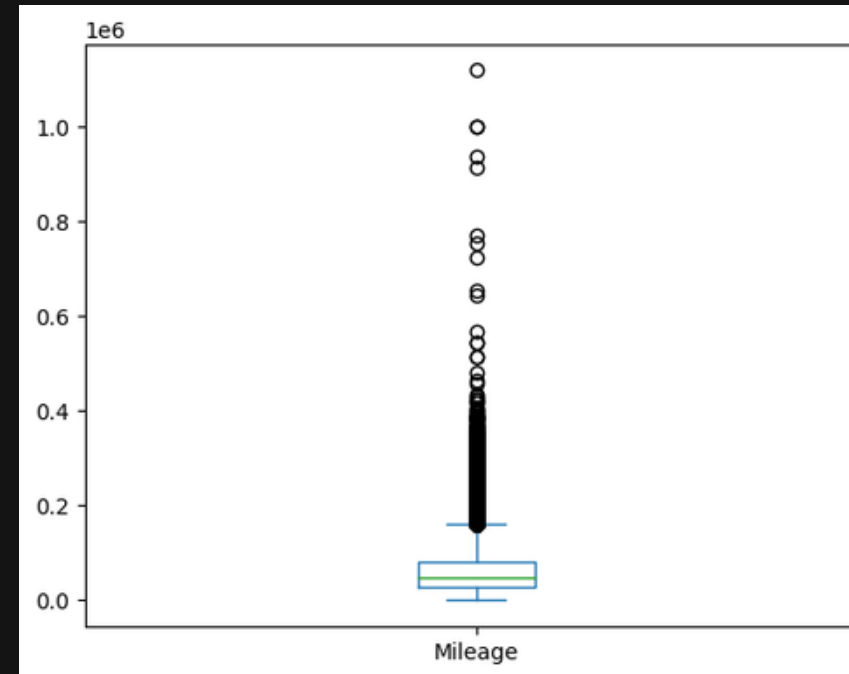
```python
1 fuel_type_mapping = {
2     'Gasoline': 'Gasoline',
3     'E85 Flex Fuel': 'Ethanol',
4     'Hybrid': 'Hybrid',
5     'Flexible Fuel': 'Ethanol',
6     'Diesel': 'Diesel',
7     'Regular Unleaded': 'Gasoline',
8     'E85 Fl': 'Ethanol',
9     'Compressed Natural Gas': 'Natural Gas',
10    'Gasoline/Mild Electric Hybrid': 'Hybrid',
11    'Automatic': 'Gasoline',
12    'Other': 'Gasoline',
13    'Premium (Required)': 'Gasoline',
14    'Gasoline Fuel': 'Gasoline',
15    'Premium': 'Gasoline',
16    'Gaseous': 'Natural Gas',
17    'Plug-In Hybrid': 'Hybrid',
18    'Hydrogen Fuel Cell': 'Hydrogen',
19    'Bio Diesel': 'Biodiesel',
20    'Natural Gas': 'Natural Gas',
21    'Biodiesel': 'Biodiesel'
22 }
23
```

```python
1 uc_df['Fuel_type'] = uc_df['Fuel_type'].map(fuel_type_mapping)
```
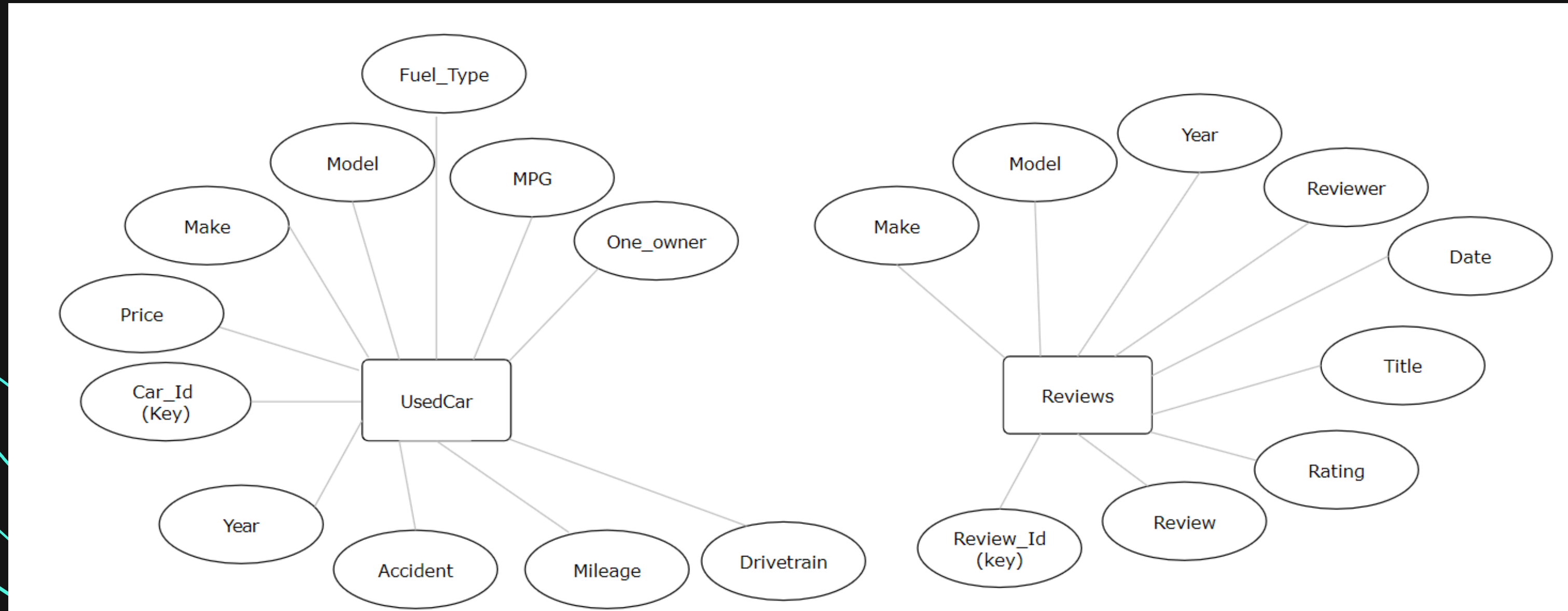

Distribution of years in Reviews
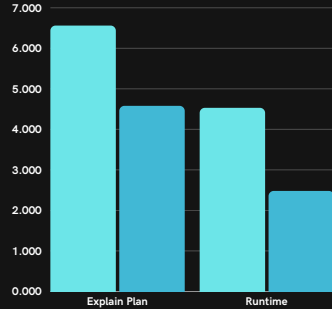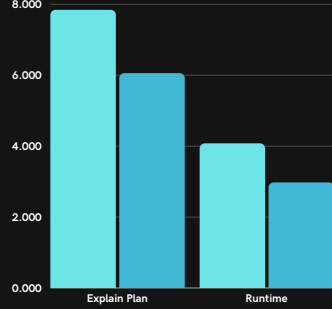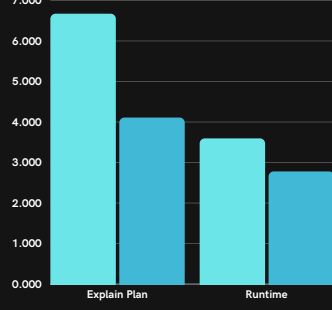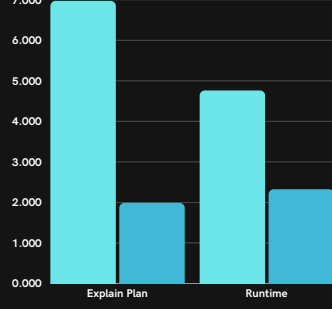
ER Diagram & BCNF

Demo

Complex Query Example

```sql
WITH RankedModels AS (
SELECT
u.Make, u.Model, AVG(r.Rating) AS AverageRating, AVG(u.Price) AS AveragePrice,
AVG(u.Mileage) AS AverageMileage, SUM(CASE WHEN u.Accident = 1.0 THEN 1 ELSE 0 END) *
100.0 / COUNT(u.Accident) AS PercentageAccidents, RANK() OVER (ORDER BY AVG(r.Rating)
DESC, AVG(u.Price)) AS Ranking
FROM UsedCars u
JOIN Reviews r ON u.Make = r.Make AND u.Model = r.Model
GROUP BY u.Make, u.Model
),
ReviewThreshold AS (
SELECT Make, Model, COUNT(*) as NumReviews
FROM Reviews r
GROUP BY Make, Model
HAVING COUNT(*) >= ${count}
)
SELECT rm.Make, rm.Model, rm.AverageRating, rm.AveragePrice, rm.AverageMileage,
rm.PercentageAccidents, rt.NumReviews, rm.Ranking
FROM RankedModels rm JOIN ReviewThreshold rt ON rm.Make=rt.Make AND rm.Model=rt.Model
WHERE rm.AverageMileage < ${avgMileage} AND rm.PercentageAccidents < ${pctAccidents}
ORDER BY rm.AverageMileage DESC, rm.PercentageAccidents;
```

Query Performance

- Used EXPLAIN PLAN to identify performance issues, used caching/"materialized views" to improve speed
- Indexes created: UsedCars(Make,Model,Year,Price), Reviews(Make,Model,Year,Rating), UsedCars(Year,Price,Mileage,MPG,Make,Model)

Query Description	Explain Plan Original Cost	Explain Plan Updated Cost	Original Runtime (ms)	Updated Runtime (ms)
Most Durable	3,640,000	38,231	33,964	304
Hidden Gems	70,000,000	1,150,000	12,225	949
Popular	4,710,000	12,822	3,943	603
Fuel Type	9,370,000	97.4	57,771	211

Extra Credit - API Usage

```
return fetch(`https://api.unsplash.com/search/photos?query=${encodeURIComponent(carData.Make + ' ' + carData.Model)}&client_id=KEY`)
```


Check out our featured car of the day:

JEEP GRAND-CHEROKEE, 2017

We used the Unsplash API

Technical Challenges

Query ideation for complex queries that would be useful to end users → Customer interviews with parents!

Understanding the datasets, cleaning, row/column removal without knowing final query ideas → Extensive EDA
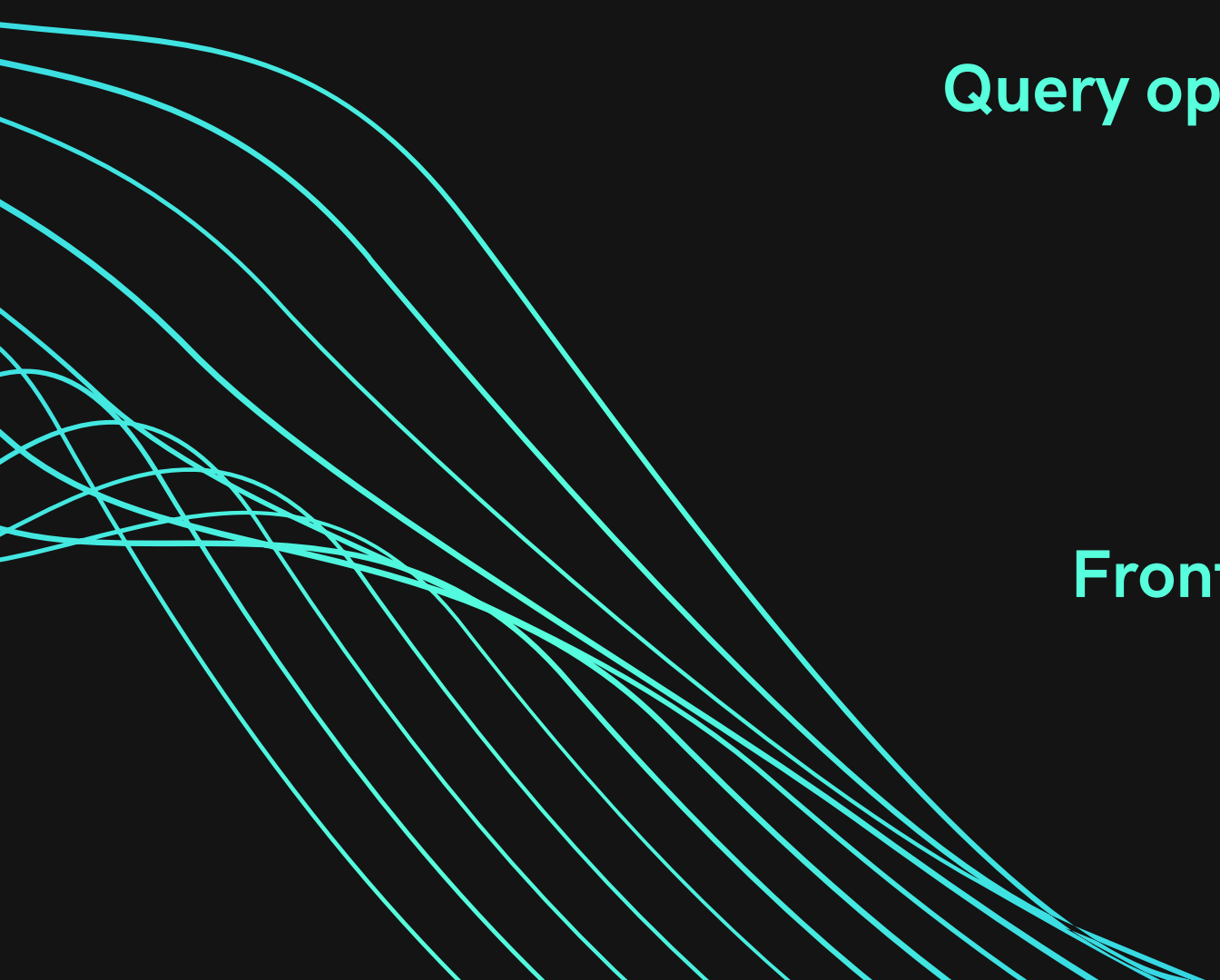
Query optimization →
- Explain Plan to identify weak points
- Caching
- Indexing
- Materialized views
- Query restructuring

Front end bugs →
- Console error log
- Stack Overflow
- And a lot of patience

Thank you, questions?