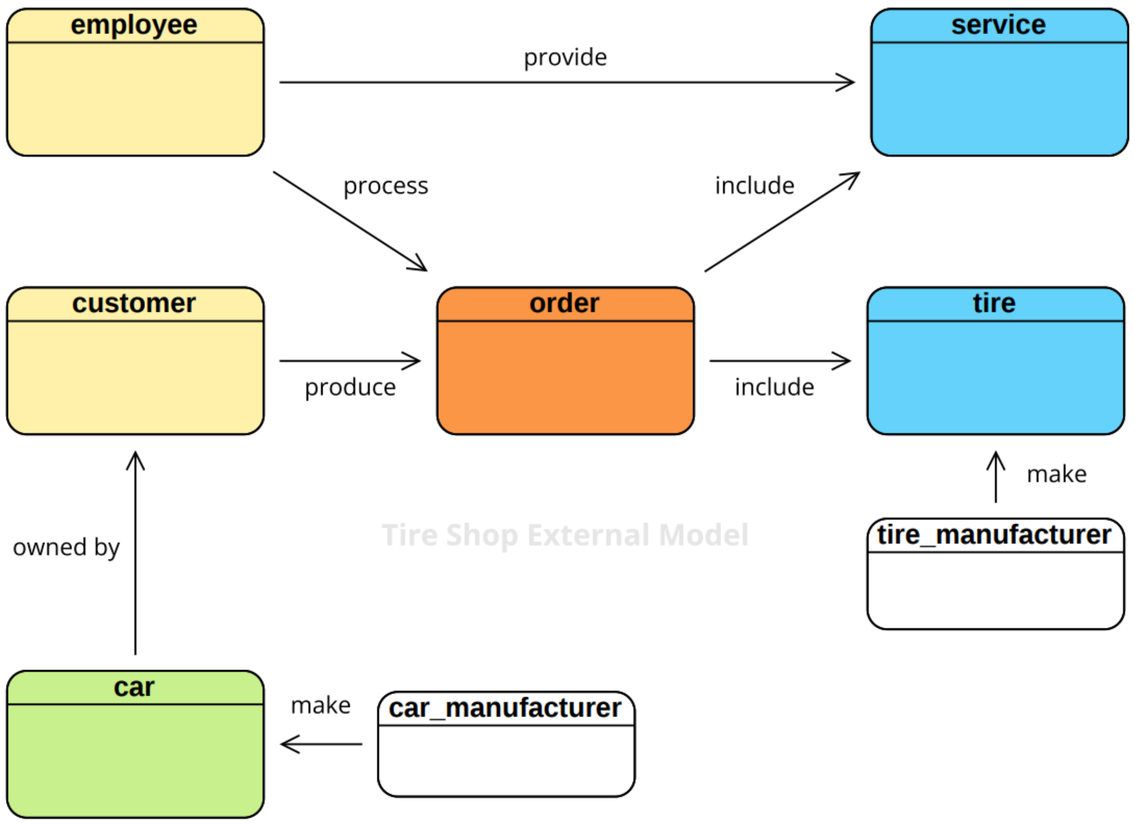
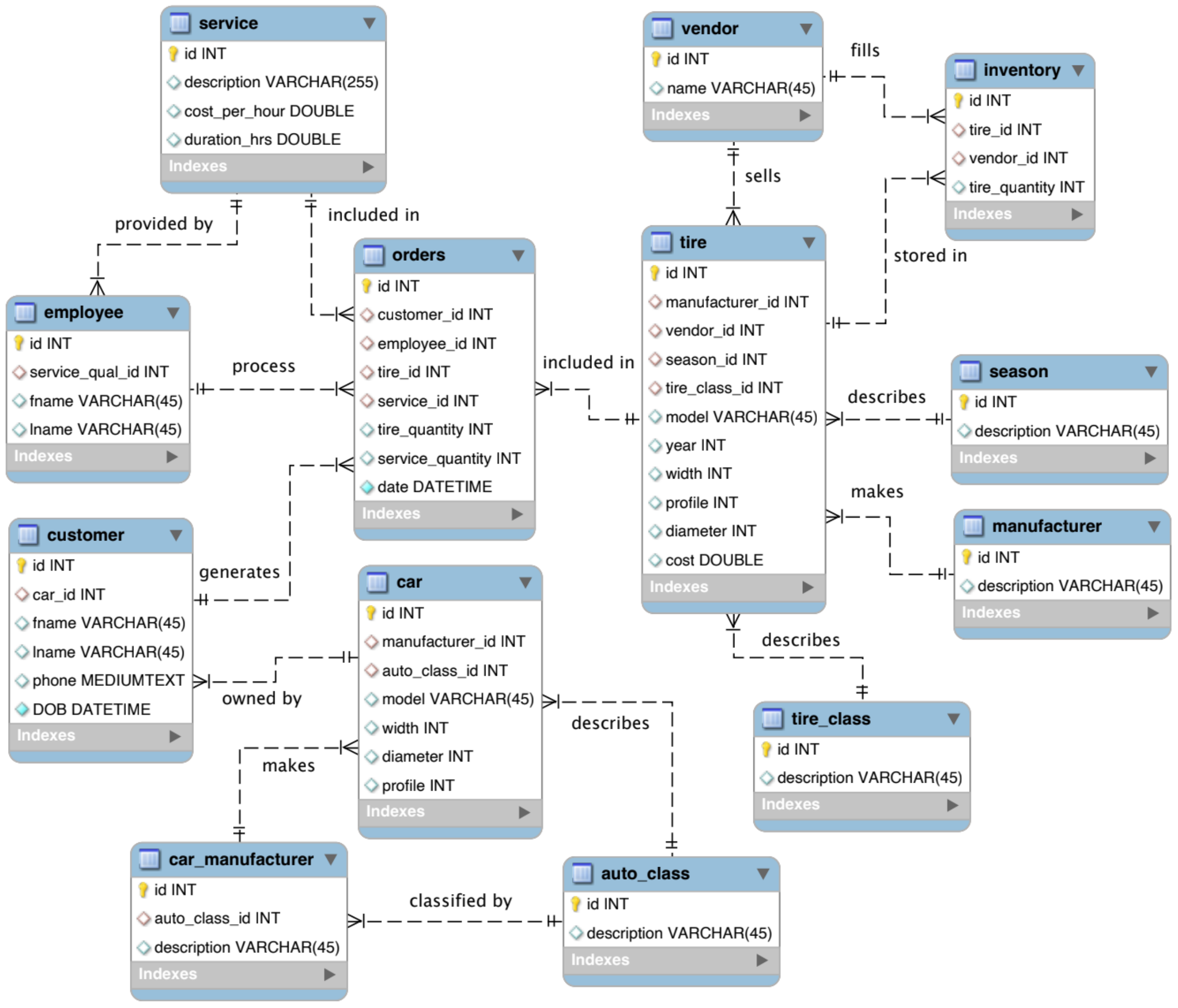
Katy Kuznetsova, Kristy Geng, Grey McBride
November 13th, 2022
CPSC 5021

# Milestone 3

1. Logical Diagram:



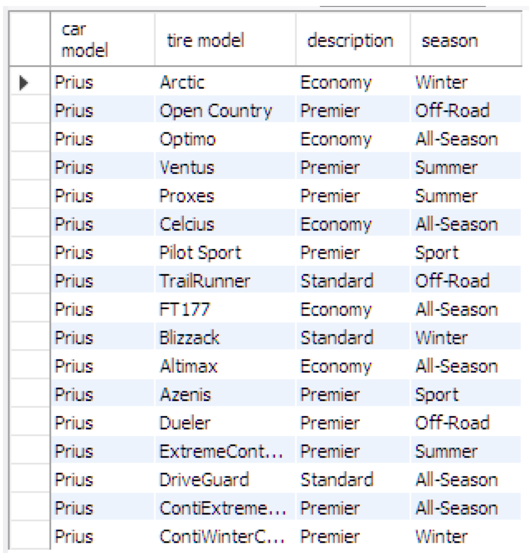
2. See SQL files (diagram from mysql)



3. Queries:

1) The following query provides an output of tire options that would fit a specific vehicle, a Prius. This query is relevant to a shop manager/employee that may be interacting with a customer that owns that particular vehicle. The result of the query provides all the options for that vehicle. I specifically included the car model column in the table in the case that a customer would look at the table, the column would provide confirmation that these are, in fact, for that vehicle.

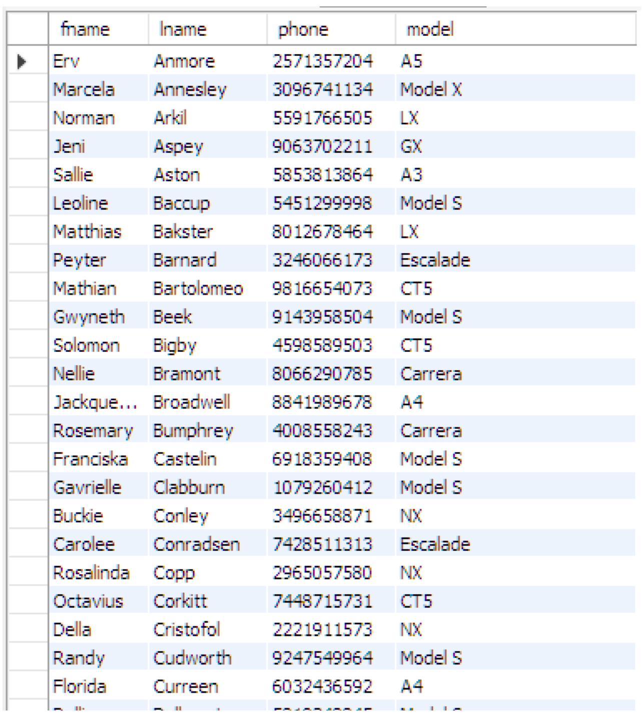
```sql
select car.model, tire.model, tire_class.description, season.description
from car
inner join tire on tire.diameter = car.diameter
inner join tire_class on tire.tire_class_id = tire_class.id
inner join season on tire.season_id = season.id
where (tire.id IN (
    SELECT id
    from tire
    where ((tire.width = car.width) && (tire.diameter = car.diameter) && (tire.profile = car.profile))
)) AND car.model = 'Prius'
ORDER BY car.id;
```



| car model | tire model | description | season |
|---|---|---|---|
| Prius | Arctic | Economy | Winter |
| Prius | Open Country | Premier | Off-Road |
| Prius | Optimo | Economy | All-Season |
| Prius | Ventus | Premier | Summer |
| Prius | Proxes | Premier | Summer |
| Prius | Celcius | Economy | All-Season |
| Prius | Pilot Sport | Premier | Sport |
| Prius | TrailRunner | Standard | Off-Road |
| Prius | FT177 | Economy | All-Season |
| Prius | Blizzack | Standard | Winter |
| Prius | Altimax | Economy | All-Season |
| Prius | Azenis | Premier | Sport |
| Prius | Dueler | Premier | Off-Road |
| Prius | ExtremeCont... | Premier | Summer |
| Prius | DriveGuard | Standard | All-Season |
| Prius | ContiExtreme... | Premier | All-Season |
| Prius | ContiWinterC... | Premier | Winter |

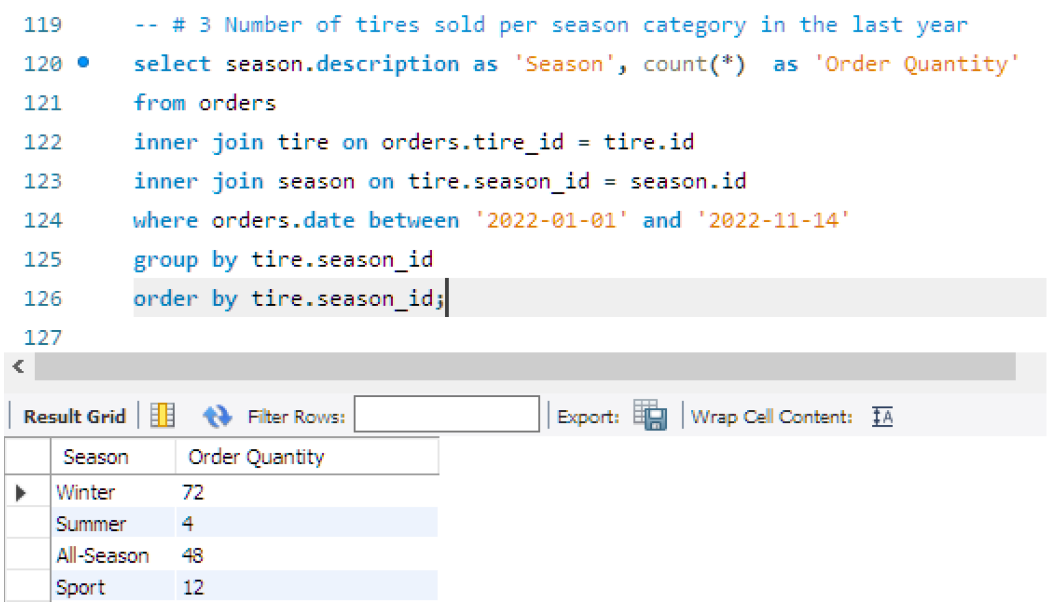
2) The following query provides the list of customers and their phone numbers that own luxury vehicles and the premier tires the store carries that fits them. This query is relevant when a store manager is planning a store sale on premier tires and wants to send a text advertisement for the sale.

```sql
select customer.fname, customer.lname, customer.phone, car.model
from customer
inner join car on customer.car_id = car.id
where car.model IN (
    select car.model
    from car
    inner join tire on tire.diameter = car.diameter
    where tire.id IN (
        SELECT id
        from tire
        where ((tire.width = car.width) && (tire.diameter = car.diameter) && (tire.profile = car.profile) &&
        ((tire.tire_class_id = 3 && car.auto_class_id = 5) ))
    )
)
ORDER BY lname;
```



| fname | lname | phone | model |
|---|---|---|---|
| Erv | Anmore | 2571357204 | A5 |
| Marcela | Annesley | 3096741134 | Model X |
| Norman | Arkil | 5591766505 | LX |
| Jeni | Aspey | 9063702211 | GX |
| Sallie | Aston | 5853813864 | A3 |
| Leoline | Baccup | 5451299998 | Model S |
| Matthias | Bakster | 8012678464 | LX |
| Peyter | Barnard | 3246066173 | Escalade |
| Mathian | Bartolomeo | 9816654073 | CT5 |
| Gwyneth | Beek | 9143958504 | Model S |
| Solomon | Bigby | 4598589503 | CT5 |
| Nellie | Bramont | 8066290785 | Carrera |
| Jackque... | Broadwell | 8841989678 | A4 |
| Rosemary | Bumphrey | 4008558243 | Carrera |
| Franciska | Castelin | 6918359408 | Model S |
| Gavrielle | Clabburn | 1079260412 | Model S |
| Buckie | Conley | 3496658871 | NX |
| Carolee | Conradsen | 7428511313 | Escalade |
| Rosalinda | Copp | 2965057580 | NX |
| Octavius | Corkitt | 7448715731 | CT5 |
| Della | Cristofol | 2221911573 | NX |
| Randy | Cudworth | 9247549964 | Model S |
| Florida | Curreen | 6032436592 | A4 |

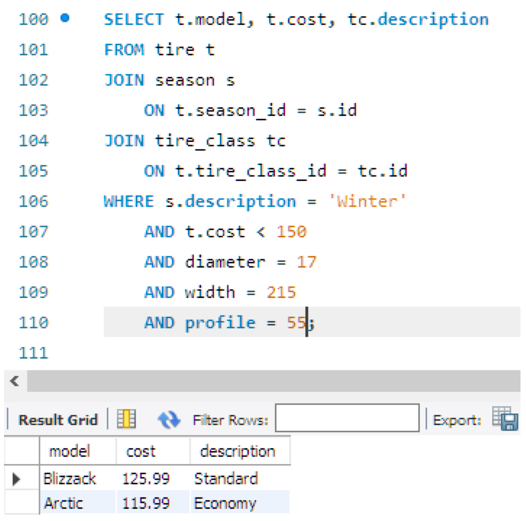
3) The following query provides the number of orders in each tire season category in the last year. This query is relevant as a use case for future business decisions such as restocking, knowing customers' interests, and knowing what tires to put on sale. From the result below, it is clear that customers buy significantly more winter and all-season tires than summer or sport tires.



```sql
-- # 3 Number of tires sold per season category in the last year
select season.description as 'Season', count(*)  as 'Order Quantity'
from orders
inner join tire on orders.tire_id = tire.id
inner join season on tire.season_id = season.id
where orders.date between '2022-01-01' and '2022-11-14'
group by tire.season_id
order by tire.season_id;
```

| Season | Order Quantity |
|---|---|
| Winter | 72 |
| Summer | 4 |
| All-Season | 48 |
| Sport | 12 |

4) Get the winter tire that cost under 150 and is a specific size. Show model, cost and tire class description. This query is relevant as a use case for when an employee is searching for a specific tire size for a customer that is budget oriented. The result would provide tire options for the customer to choose from.

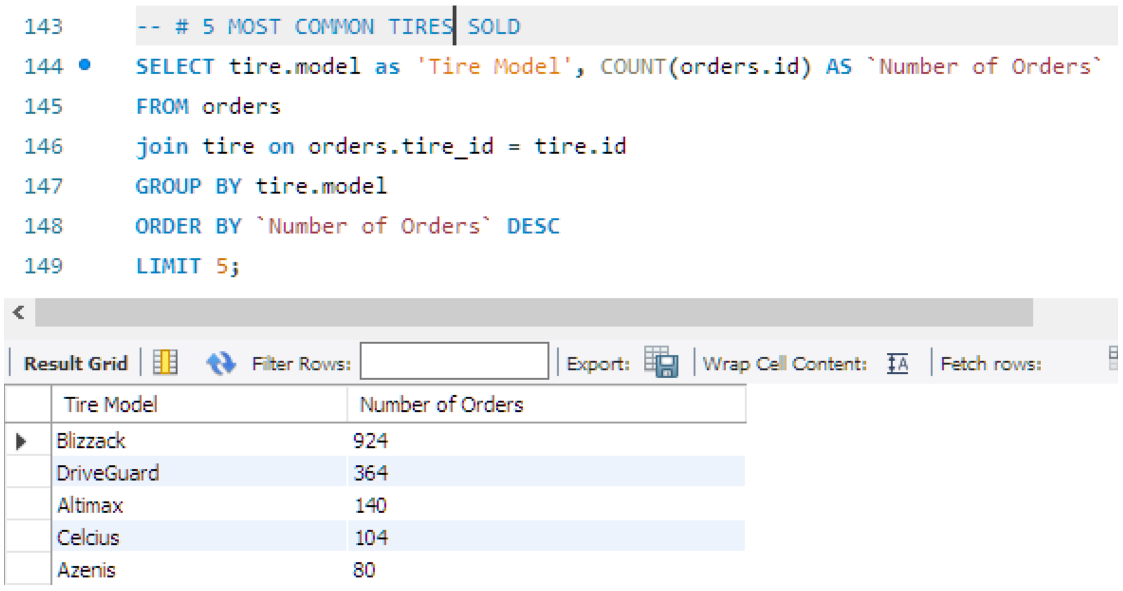


```
SELECT t.model, t.cost, tc.description
FROM tire t
JOIN season s
    ON t.season_id = s.id
JOIN tire_class tc
    ON t.tire_class_id = tc.id
WHERE s.description = 'Winter'
    AND t.cost < 150
    AND diameter = 17
    AND width = 215
    AND profile = 55;
```

Result Grid | Filter Rows: | Export:

| model | cost | description |
|---|---|---|
| Blizzack | 125.99 | Standard |
| Arctic | 115.99 | Economy |

5) This query provides the names of the top five most common tires sold and the number of orders of that tire, which is relevant to a business when deciding which tires to stop carrying in inventory and which tires should always be in stock based on customer preference.

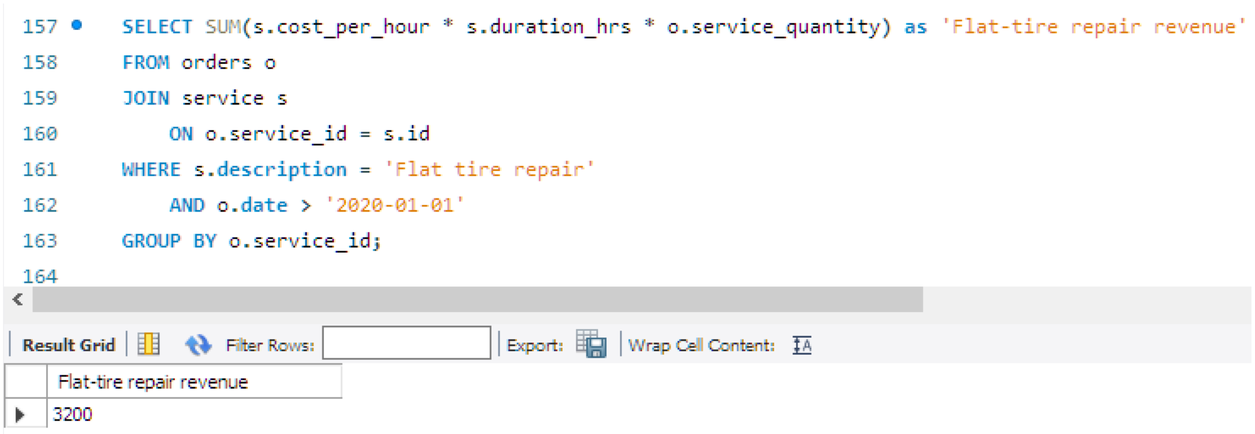


```
-- # 5 MOST COMMON TIRES SOLD
SELECT tire.model as 'Tire Model', COUNT(orders.id) AS `Number of Orders`
FROM orders
join tire on orders.tire_id = tire.id
GROUP BY tire.model
ORDER BY `Number of Orders` DESC
LIMIT 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

| Tire Model | Number of Orders |
|---|---|
| Blizzack | 924 |
| DriveGuard | 364 |
| Altimax | 140 |
| Celcius | 104 |
| Azenis | 80 |

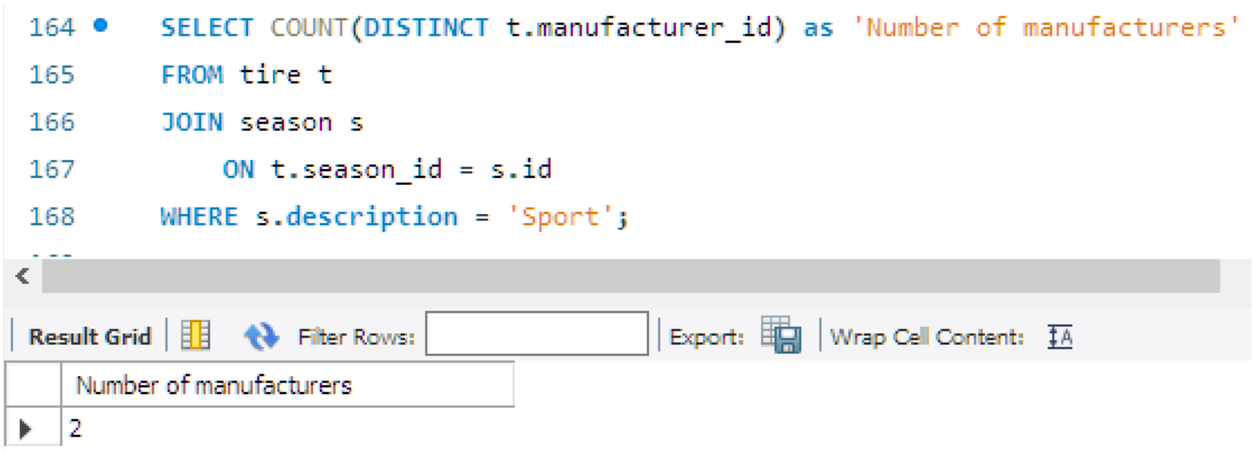
6) The following query results in revenue the store generated by doing flat-tire repairs for customers. This is a relevant query for making the business decision if the service is generating enough revenue to continue doing this service.



```sql
SELECT SUM(s.cost_per_hour * s.duration_hrs * o.service_quantity) as 'Flat-tire repair revenue'
FROM orders o
JOIN service s
    ON o.service_id = s.id
WHERE s.description = 'Flat tire repair'
    AND o.date > '2020-01-01'
GROUP BY o.service_id;
```

```
Flat-tire repair revenue
3200
```

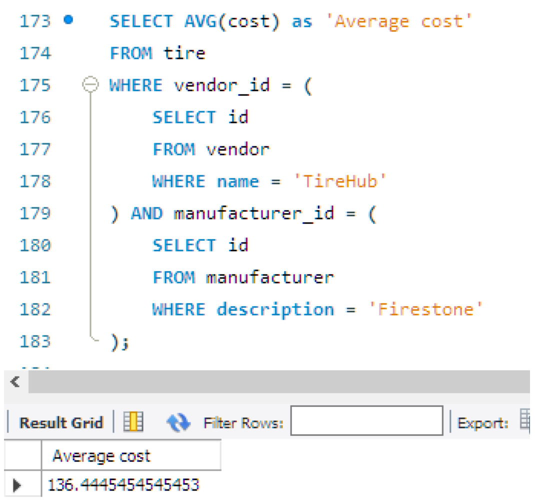
7) This query would provide the store manager the number of manufacturers in the store system provide sport tires. This information could be relevant if store staff is looking an alternative option for a type of tire or if they have options for sport tires if a manufacturer stops producing a sport tire. This type of information allows a business' inventory to be robust in cases of supply shortages.



```sql
SELECT COUNT(DISTINCT t.manufacturer_id) as 'Number of manufacturers'
FROM tire t
JOIN season s
    ON t.season_id = s.id
WHERE s.description = 'Sport';
```

```
Number of manufacturers
2
```

8) This query results in the average cost of Firestone ( example manufacturer) tires provided by TireHub (example vendor). This information might be valuable for store managers when comparing prices of tires between different vendors or between different manufacturers.

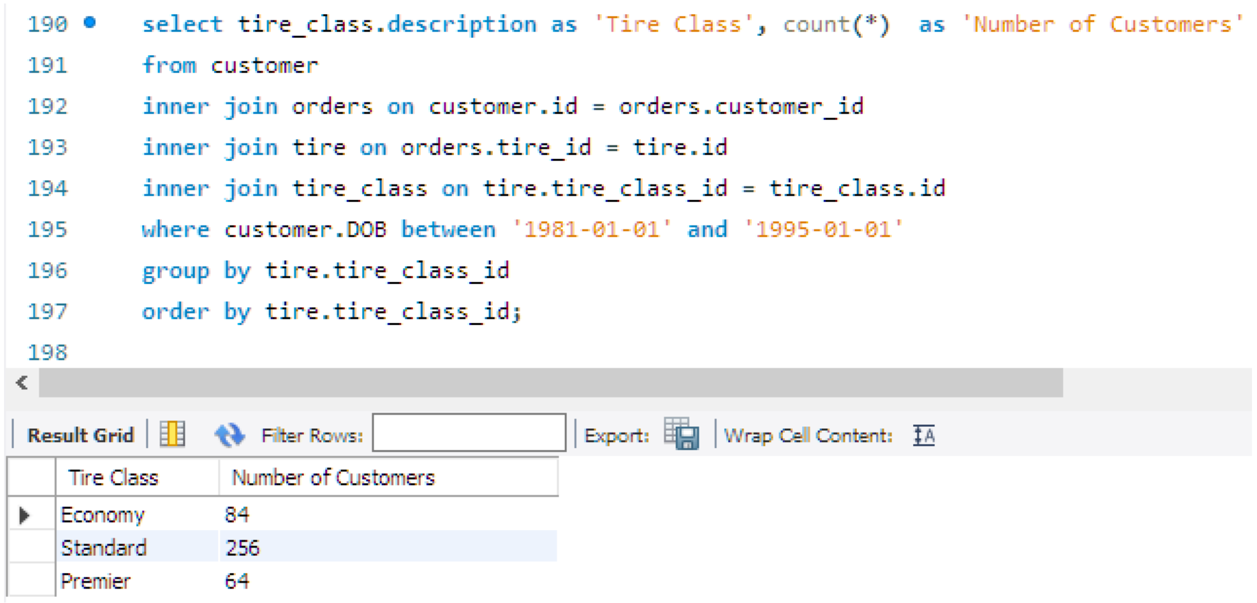


```
SELECT AVG(cost) as 'Average cost'
FROM tire
WHERE vendor_id = (
    SELECT id
    FROM vendor
    WHERE name = 'TireHub'
) AND manufacturer_id = (
    SELECT id
    FROM manufacturer
    WHERE description = 'Firestone'
);
```

Result Grid | Filter Rows: | Export:

| Average cost |
|---|
| 136.4445454545453 |

9) This query shows the number of customers that fall in the millennial generation and the tire class they choose to purchase. This result is relevant because it allows the business to understand its customer base better. This shows that millennials tend to buy the standard tires far more than the cheapest or most expensive options. Information like this can help with marketing and future product decisions.



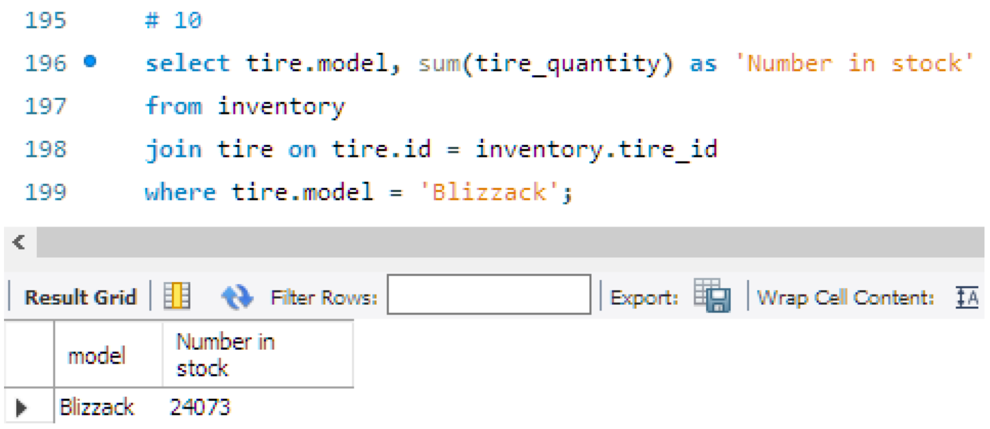
```
select tire_class.description as 'Tire Class', count(*)  as 'Number of Customers'
from customer
inner join orders on customer.id = orders.customer_id
inner join tire on orders.tire_id = tire.id
inner join tire_class on tire.tire_class_id = tire_class.id
where customer.DOB between '1981-01-01' and '1995-01-01'
group by tire.tire_class_id
order by tire.tire_class_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| Tire Class | Number of Customers |
|---|---|
| Economy | 84 |
| Standard | 256 |
| Premier | 64 |

10) This query provides the number of tires in stock of a specific tire model. A query like this is relevant when a store manager/employee needs to know if a certain tire needs restocking. In this case, I ran the query with the most common tire sold

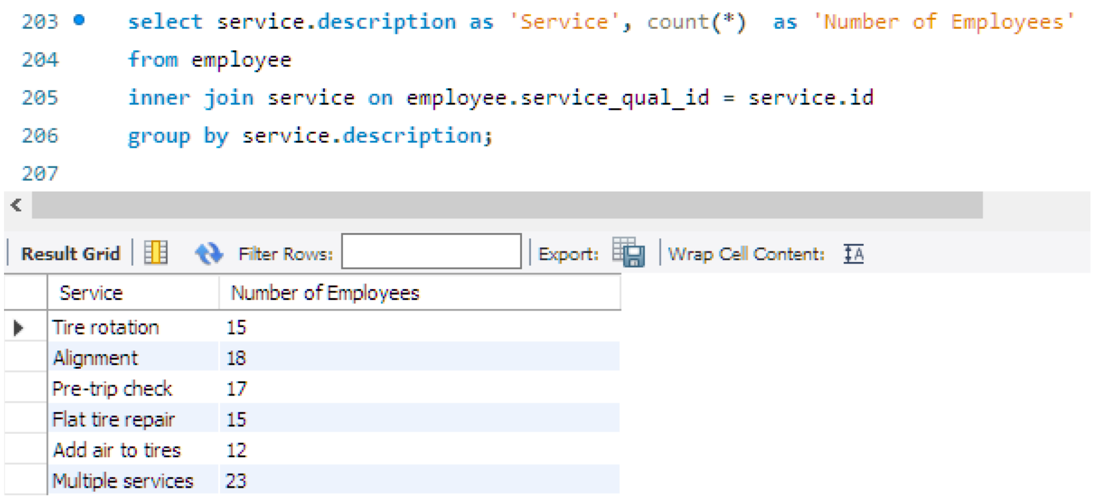
(information found by an earlier query), a product that the store would not want to be low on due to its popularity.



```
# 10
select tire.model, sum(tire_quantity) as 'Number in stock'
from inventory
join tire on tire.id = inventory.tire_id
where tire.model = 'Blizzack';
```

| model | Number in stock |
|---|---|
| Blizzack | 24073 |

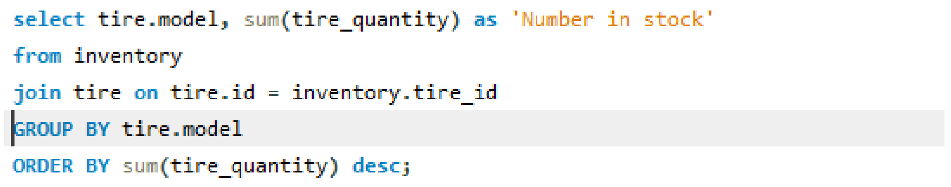
11) This query shows the number of employees qualified to do each of the services that the shop provides. This result/information is relevant for creating employee schedules, deciding if additional hiring is needed, or finding if training is needed for a particular skill for the employees.



```
select service.description as 'Service', count(*)  as 'Number of Employees'
from employee
inner join service on employee.service_qual_id = service.id
group by service.description;
```

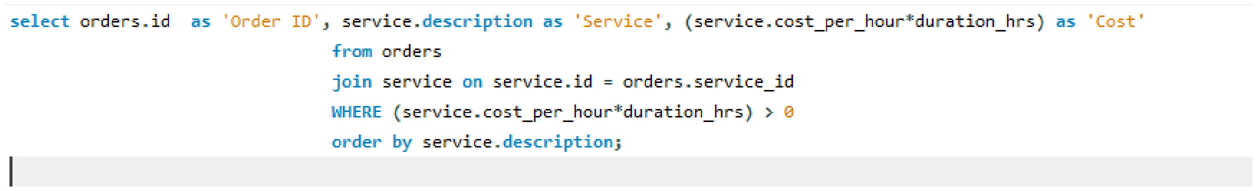
| Service | Number of Employees |
|---|---|
| Tire rotation | 15 |
| Alignment | 18 |
| Pre-trip check | 17 |
| Flat tire repair | 15 |
| Add air to tires | 12 |
| Multiple services | 23 |

12) This query shows the number of tires for each model of tire that are currently in the store's inventory. This is a very basic but essential query to be used in any sort of inventory management or ordering system to ensure you know what is currently in stock.



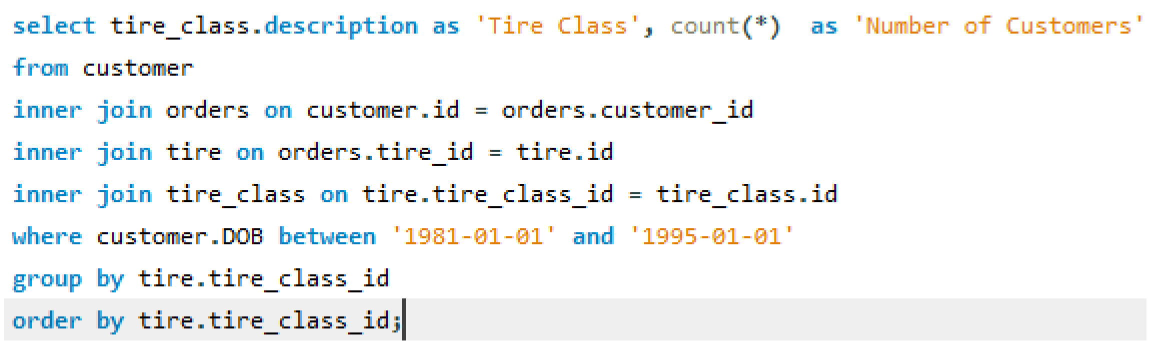
```
select tire.model, sum(tire_quantity) as 'Number in stock'
from inventory
join tire on tire.id = inventory.tire_id
GROUP BY tire.model
ORDER BY sum(tire_quantity) desc;
```

13) This query selects all orders that include services, and displays the cost of the service as well as which service was provided. This would be a useful query to check on which services are being done, as well as to verify cost for a previous customer. It would also be useful in determining price points for services and whether the cost per hour is working out as intended.



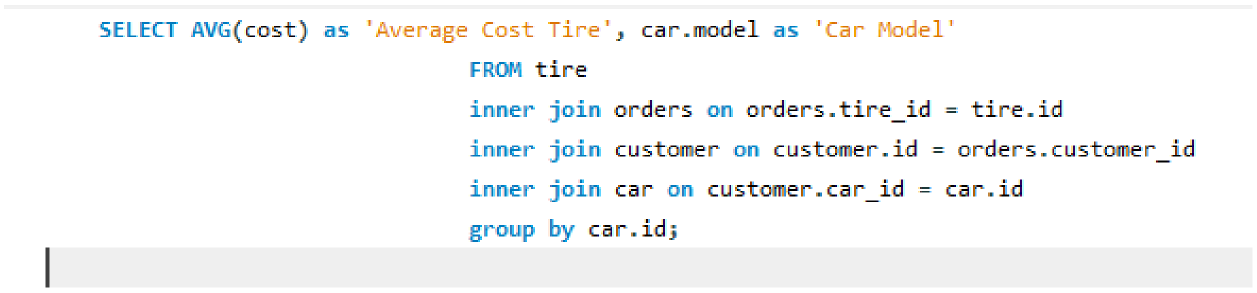
```
select orders.id  as 'Order ID', service.description as 'Service', (service.cost_per_hour*duration_hrs) as 'Cost'
                                from orders
                                join service on service.id = orders.service_id
                                WHERE (service.cost_per_hour*duration_hrs) > 0
                                order by service.description;
```

14) This query displays the most popular choice of tire class for a given generation / age-range of customers (example millennials). This would be a useful query to identify trends in customer opinions and could affect decisions for the business, as well as assist employees in making sales by recommending tires that a given customer is more likely to want to purchase.



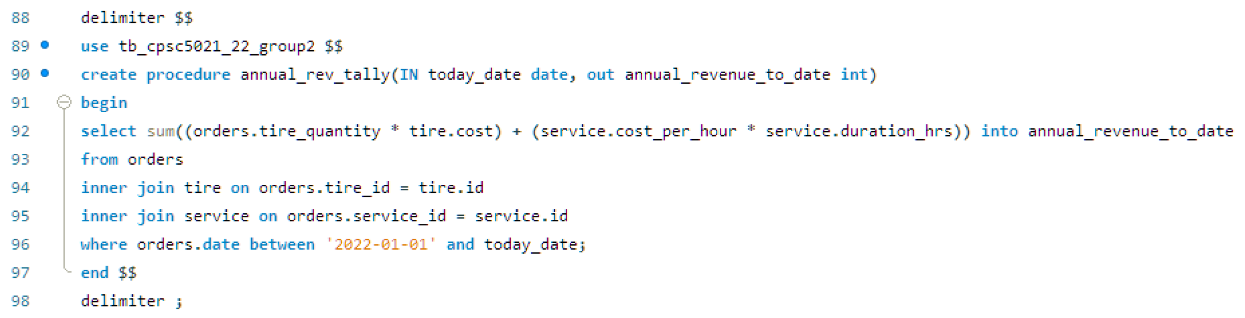
```
select tire_class.description as 'Tire Class', count(*)  as 'Number of Customers'
from customer
inner join orders on customer.id = orders.customer_id
inner join tire on orders.tire_id = tire.id
inner join tire_class on tire.tire_class_id = tire_class.id
where customer.DOB between '1981-01-01' and '1995-01-01'
group by tire.tire_class_id
order by tire.tire_class_id;
```

15) This query displays the average cost of tires for each car model. This would be a useful query in terms of making sales to customers and providing them with an informed decision as to what it will cost based on just their car model. It would also be a great tool for seeing which car models are the most grossly profitable for the business which would be used to make informed decisions about pricing, products, and overall business strategies.

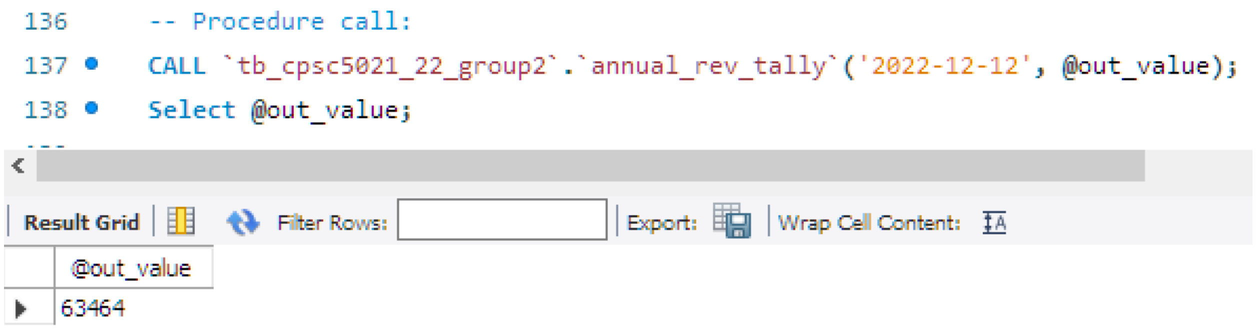


```
    SELECT AVG(cost) as 'Average Cost Tire', car.model as 'Car Model'
                            FROM tire
                            inner join orders on orders.tire_id = tire.id
                            inner join customer on customer.id = orders.customer_id
                            inner join car on customer.car_id = car.id
                            group by car.id;
```

4. Procedure to sum the total revenue from the start of the year to the date entered into the procedure. The output is the result of the sum of the products of tire/service costs and their quantities. This type of procedure would be useful when a store manager is creating a possible annual sales report.



```
delimiter $$
use tb_cpsc5021_22_group2 $$
create procedure annual_rev_tally(IN today_date date, out annual_revenue_to_date int)
begin
select sum((orders.tire_quantity * tire.cost) + (service.cost_per_hour * service.duration_hrs)) into annual_revenue_to_date
from orders
inner join tire on orders.tire_id = tire.id
inner join service on orders.service_id = service.id
where orders.date between '2022-01-01' and today_date;
end $$
delimiter ;
```



```
-- Procedure call:
CALL `tb_cpsc5021_22_group2`.`annual_rev_tally`('2022-12-12', @out_value);
Select @out_value;
```

```
@out_value
63464
```

5. Link to Google Slideshow (public permissions):

   https://docs.google.com/presentation/d/1S8bnNr-UecHVlaNDrm_Rw25n266EvSbZ50NK2tKKD54/edit?usp=sharing