# COMP207 – Assignment 1: The Epature bus company database

There are 100 points you can get in this assignment. Recall that the assignment contributes 25% to your overall grade in the course.

**The assignment is to create a database for a bus company called Epature:** The initial task will focus on building the database (i.e., tables and so on), while the latter tasks focus on creating useful queries.

Questions 2-7 give some number of points each as is marked on them.

- **Public data:** For each question, 2 points are given for getting the right output on the public test data described in an additional file. The question will specify what this right output is, and you will also be able to see it when you upload your solution to CodeGrade (since you can upload any number of times before the deadline, I would suggest using it to test your solution against the test data)
- **Hidden data:** The remaining points are given for getting the right output on another data set. The latter set is kept hidden to avoid you hardcoding the right output in the queries. It will follow the same form as the test data though and unless you hardcode your solution for the test data, it is very likely that if you get points for one part you will get points for the other. (Note that you do get the grade at once for task 1).

Because I won't have much time to help each of you individually (I think there will be around 550 of you so you do the math), I would prefer that you put any questions you had on the discussion board. That said, **you are NOT meant to include your code on the discussion board** – simply reference the code you uploaded to CodeGrade if needed. This also means that I would prefer that you try first on your own and/or check the discussion board (because with so many of you, most questions will have been asked and answered at the time most of you think of them and it is faster to look it up for you than waiting for me to answer it again!) and then ask.

**How much help you can expect if you ask for it**: I am willing to help you a lot if you get stuck in questions 1-2 (but please, spend some time on them first – nearly everybody – think >95% - won't have much trouble with them). For questions 3-5 I will come up with suggestions and ideas to help guide you to a solution (but again, please try first! The questions already have some hints so check those first). Questions 6-7 are meant to be hard. If you do the questions in order, these questions will get your grade from 80 to 100. According to the university's marking scheme, you need to be able to answer every question perfectly to get a grade in the range 70-80(!). Therefore, I will only help clarify what the questions are asking you to do but not actually help you solve them.

Each question from 2-7 requires you to create a view with a specified (in the question) name. You may use as many views as you wish to solve each question (well, except question 1, since it would not be helpful), but there should be one with the specified name, which is the one that is getting checked for having the right output. E.g., you could have **view1ForQuestion2**, **view2ForQuestion2** and **view3ForQuestion2**, if you feel it would help to do question 2 – most other names are fine too!

**ChatGPT:** You are not allowed to use ChatGPT.

**Group work:** While you might discuss with your friends/colleagues, you must hand in a solution only you worked on, and plagiarism checks will be run.

## Deadline and feedback

The deadline for the assignment is **Wednesday the 8th of November at 17:00**. General feedback for the assignment will be given on Tuesday the 21st of November. The relatively long period between those is because you can get an ELP and they only last until I give feedback. To accommodate people with ELPs as well as I can, I will therefore first give feedback nearly 2 weeks after the deadline (2 weeks is the maximum amount of time an ELP can give you).

If you have specific concerns about your grade or similar for the first assignment, then, after the general feedback has been released, I will answer questions about your solution and grade over email.

## Format

**The assignment should be done in .sql format** (i.e., the output format from MySQL's workbench) – it is really just a basic text file with the SQL commands written in it and you could do it by writing the file directly in a basic text editor if you wish (Notepad in Windows or TextEdit on Mac – if you select Make Plain Text in Format).

**The name of the file should be epature.sql**: You can hand in precisely 1 file, and it must have precisely that name (you can submit as many times as you wish until the deadline, but only the most recent version counts).

**And each line should contain only the following:**

1. CREATE TABLE statements for question 1 (6 in total)
2. CREATE VIEW statements for questions 2-7 (the number of views depends on how you solve the questions and how many you solve, if not all). Note, that you may use any positive number of views to solve each question, but each question's specified view should have the properties requested.
3. SQL comments, i.e. the part of lines after "-- ", i.e. double - followed by space. You do not need to make any, but may do so if you wish.

In particular, **do not include CREATE DATABASE and USE statements**. They will make the tests on the hidden data not work (technically, I create two databases based on your construction, one with the public data given in the additional file and one with the hidden data. If you use CREATE DATABASE or USE statements, in essence only 1 is made and it will be marked as if you did not do the other one). I have written tests that test for this so if you do not do something really complex to avoid these, you should at least be warned about it.

You can include INSERT statements, but the database will be emptied before checking, so it serves little purpose (often, many people will submit with the test data already inserted, but, because the databases will be emptied, it will not matter).

It is very unlikely that you would want to remove any views after you have made them. Unless you have a very good reason, do not remove them.

**Make sure that you can run the full file through MySQL** when using the Epature database (starting with an empty Epature database) and after having done so, the Epature database should contain the tables and views required from the questions you solved (and perhaps some more views if you feel it would be convenient). This means that **you should remove any statement that causes errors before handing in the assignment,** because MySQL stops when it encounters an error (meaning that the last statements are not executed)! If you do not, you risk getting a far lower grade than otherwise (because the part of your hand-in after the first error will not be graded).

**You can submit any number of times before the deadline:** We are using CodeGrade for checking these things and whenever you submit, you will see whether your file works for the public dataset. I suggest using it…

**Do *not* do the following:** Any of the following should ***not*** be done:

- End by removing the database (i.e. DROP DATABASE Epature; or similar). It would be the same as handing in an empty file. It will be very easy to see on CodeGrade since everything will stop working.
- Create comments like "------------". MySQL Workbench will accept it, but the command line version of MySQL does not, which is what is used to check your file... Just insert an extra space after the second -.
- Use any other order for the columns than what is specified. Since the insert command does not state which columns they insert into, you will put the information in the wrong column and then get hard-to-understand issues when you attempt to solve the questions.
- Use PARTITION OVER or other new commands. It was not taught in class and is specific to newer versions of MySQL (like the one you would install on your own laptop). Unfortunately, CodeGrade is using an old version of Ubuntu (or at least did last year), where the newest versions of MySQL do not work. Therefore, it will not work when we grade you and you will fail that (and later questions – MySQL will report an error on that line and will not run the rest of your file).

## Question 1 – (Easy) (worth 18 points – 3 points for each table)

Make the set of tables that match the following set of schemas.

- **Customers**(birth_day, first_name, last_name, c_id)
- **Employees**(birth_day, first_name, last_name, e_id)
- **BusType**(capacity,type)
- **BusTrip**(start_time,route_no,type*,e_id*,b_id)
- **TicketCosts**(cost,duration)
- **CustomerTrip**(from_stop_no,to_stop_no,b_id*,c_id*)

Each underlined attribute should be the primary key for the table (it happens to be the last attribute in each table besides CustomerTrip where it is the last two together) and each attribute with * should have a foreign key to the table with a primary key of just that name, e.g. if the tables were R(a,b) and S(b*,c), b in R and c in S should be the primary keys and b in S should reference b in R as a foreign key. More directly, type and e_id in BusTrip and b_id and c_id in CustomerTrip should reference the primary keys of BusType, Employees, BusTrip and Customers respectively.

Only use data types in the following list: INT, VARCHAR(20), DATE, DATETIME. Instead of specifying the datatypes explicitly, ensure that the test data defined in the additional file gets inserted correctly (it seems very likely that you would also guess the same datatypes as these suggest – except perhaps duration which is a VARCHAR(20) – it will be one of the strings single, day or week) and use DATE or DATETIME if all entries are dates or date-times. If you follow all of these requirements, each attribute should have a clear, unique datatype (which happens to likely be what you would guess it to be). As an aside, the costs are measured in pennies (and not directly pounds), to avoid precision issues with floating point numbers.

## Question 2 – (Easy)

(worth 21 points – 2 points for getting the right output on the test data and another 20 for the hidden data – see the first page for more detail!)

We are considering giving Louise Davies a raise but want to check how many bus trips she has made in September 2023 first. More precisely, you are asked to create a view **LouiseTrips** with number_of_trips which should be how many trips were done by Louise Davies in September 2023 (you may assume she did some and that she is the only employee with that full name – you should **NOT** assume that her employee id is 4 just because it is in the public data!).

Note that in the test data, Louise did 11 trips, of which 8 where in September 2023, 1 in September 2022, 1 in August 2023 and 1 in October.

HINT: Recall that COUNT(*) will count how many rows you have in the output, so you just need to make a query that finds the bus trips that were done in September 2023 by Louise Davies and then use that.

The view should be called **LouiseTrips** and be such that the output of

SELECT * FROM **LouiseTrips**;

when run on the Epature database (after inserting the test data given in an additional file) should be:

| number_of_trips |
|---|
| 8 |

## Question 3 – (Easy-medium)

(worth 15 points – 2 points for getting the right output on the test data and another 13 for the hidden data – see the first page for more detail!)

The bus company want to change route 102 but wants to hear from employees and customers who have been on that route first. You are meant to help with that and thus should find the customers and employees that have been on route 102 and you should return the birth_day, first_name,last_name of those people. HINT: You should likely use UNION.

The view should called be **PeopleOnRoute** and be such that the output of

SELECT * FROM **PeopleOnRoute** ORDER BY last_name, first_name;

when run on the Epature database (after inserting the test data given in an additional file) should be:

| birth_day | first_name | last_name |
|---|---|---|
| 1992-12-17 | Sofia | Adams |
| 1989-12-31 | Chloe | Allen |
| 1986-03-15 | Christopher | Anderson |
| 1985-10-23 | Avery | Baker |
| 1992-09-05 | Emily | Brown |
| 1984-09-02 | Ava | Clark |
| 1992-11-10 | Louise | Davies |
| 1991-02-18 | Sarah | Davis |
| 1994-09-29 | Ethan | Green |
| 1988-06-19 | Lily | Hall |

| | | |
|---|---|---|
| 1996-08-17 | Andrew | Harris |
| 1985-11-25 | David | Jones |
| 1993-04-06 | Jacob | King |
| 1990-11-11 | William | Lewis |
| 1988-04-30 | Matthew | Miller |
| 1989-10-12 | Daniel | Moore |
| 1987-01-04 | Mia | Robinson |
| 1986-07-07 | Grace | Scott |
| 1985-08-20 | Jane | Smith |
| 1990-01-15 | John | Smith |
| 1994-12-08 | Sophia | Taylor |
| 1993-05-28 | Emma | Thomas |
| 1995-02-25 | James | Walker |
| 1987-06-10 | Michael | Williams |
| 1997-07-22 | Olivia | Wilson |
| 1991-08-14 | Benjamin | Young |

(it is Jane Smith and Louise Davies as employees and all customers besides Alice Smith)

## Question 4 – (Medium) (worth 15 points – 2 points for getting the right output on the test data and another 13 for the hidden data – see the first page for more detail!)

A manager wants to be able to wish each employee happy birthday on their birthdays. Output all the employees (birth_day,first_name,last_name) sorted by how far in the future their birthday is, from the 8th of November (i.e. the day you are meant to hand in) – so any on the 8th of November would be the first, followed by any on the 9th of November and so on and any on the 31st of December would be before any on the 1st of January and so on. Finally, any on the 7th of November would be last.

HINT: Given a date D, you can get which month it is in using MONTH(D) and the day it is in that month using DAYOFMONTH(D). There is a similar function for the day in the year, but if you used that you run into issues with people born in a leap years (specifically birthdays in March or later would be off-by-1). One way to do the query is to find the people that still have their birthday this year in some view where you make a constant 1 in a new column and then find the ones that first have their birthday next year and give them the constant 2 in the same new column and putting those together with UNION. You can then order the UNION by the new column, the month and the dayofmonth of the birthdays.

Note, your query is NOT meant to use the current date, but specifically the 8th of November – it is likely easy to convert it to the current date, using CURDATE() but it would make the query change depending on the day it gets checked which would be annoying for me.

The view you create should be called **UpcomingBirthdays** and it should be such that

SELECT * FROM **UpcomingBirthdays**;

when run on the Epature database (after inserting the test data given in an additional file) should be:

| birth_day | first_name | last_name |
|---|---|---|
| 1992-11-10 | Louise | Davies |
| 1988-12-22 | Emily | Doe |
| 1995-04-10 | Alice | Johnson |
| 1990-05-15 | John | Doe |
| 1985-08-20 | Jane | Smith |

## Question 5 – (Medium-Hard)

(worth 10 points – 2 points for getting the right output on the test data and another 8 for the hidden data – see the first page for more detail!)

We want to ensure that the buses do not have too many passengers at any time. For each bus trip, i.e. b_id, determine if there is a stop_no so that the number of passengers that were on the bus at that time, (i.e. passengers in **CustomerTrip** so that from_stop_no<=stop_no<to_stop_no) is more than the bus capacity (in **BusType**). You can assume that the buses are empty at the start of the trip and at the end (i.e. for each bus trip in **BusTrip**).

HINT: You can do it with just 1 SELECT query fairly directly, but you will need to use WHERE, GROUP BY, HAVING in that case and need to have **CustomerTrip** on the FROM line <u>twice</u> (and **BusType** and **BusTrip** once). That said, it might be conceptually easier to understand if you made a view that gives the stop_no's (i.e. both from_stop_no and to_stop_no in one column – that said, if you think about it, you only need to check for whether the bus is full on a from_stop_no – if none got on at a given stop, it can only be overfull if it was overfull already, so you really only need from_stop_no) and use that instead of one of the CustomerTrips (you will still need WHERE, GROUP BY, HAVING).

The view you create should be called **OverfullBuses** and it should be such that

SELECT * FROM **OverfullBuses** ORDER BY b_id;

when run on the Epature database (after inserting the test data given in an additional file) should be:

| b_id |
|---|
| 27 |
| 28 |
| 42 |
| 44 |
| 49 |
| 50 |

## Question 6 – (Hard)

(worth 10 points – 2 points for getting the right output on the test data and another 8 for the hidden data – see the first page for more detail!)

The bus company wants to use a scan and drive system: When you, as a customer, go on the bus, you scan your bank card and then end up having to pay for the cheapest tickets that could be used for your trips – whether they are single, day or weekly tickets.

To make it slightly easier to solve, this and the next question will together solve that.

Here, in this question, we will only look for the best tickets between single and day tickets.

For each date in which the customer took at least one trip, determine what the customer should pay for all their trips that day in total (you can see the prices of the different ticket types in **TicketCosts**). The output should be c_id for the customer, date for the date and cost for the total cost.

As an example, if price for a single ticket is 200 and the price for a day ticket is 500, then, if you went on two trips that day, you would pay 200*2=400, while if you went on three or more you would pay 500.

To make it a bit easier, the buses stop before midnight. Given a datetime D, you can get the corresponding date using the function DATE(D)

The view should be called **PricePerDay.** Because the output is very large on the example database, it is presented at the end, from pages 13 to 19.

## Question 7 – (Hard) (worth 10 points – 2 points for getting the right output on the test data and another 8 for the hidden data – see the first page for more detail!)

Note, that this question is a continuation of question 6.

You are meant to extend your solution in question 6 to also handle weekly tickets. You are meant to output c_id, week, year, cost for the cost for that week in that year for that customer.

To make it easier, weekly tickets run from Monday to Sunday. Also, to make it less tedious you may assume that none takes the bus during the last and first week of the year.

HINT: You can find which week a datetime D is in using WEEK(D,7) – the 7 is for Monday. Also, you can find the year using YEAR(D).

E.g., if the weekly ticket cost was 1600, the daily ticket cost was 500 and the single ticket cost was 200, then it would be cheaper to pay 3 day tickets if you went on 3 trips on each of those days (1500), when one weekly ticket (1600). On the other hand, it would be better to buy a weekly ticket than buying 2 single tickets each day (2800 vs 1600).

The view should be called **PricePerWeek** and be such that the output of

SELECT * FROM **PricePerWeek** ORDER BY c_id, week, year;

when run on the Epature database (after inserting the test data given in an additional file) should be:

| c_id | week | year | cost |
|---|---|---|---|
| 1 | 20 | 2022 | 200 |
| 1 | 35 | 2021 | 200 |
| 1 | 35 | 2023 | 200 |
| 1 | 36 | 2023 | 200 |
| 1 | 37 | 2023 | 1300 |
| 1 | 38 | 2023 | 200 |
| 1 | 51 | 2022 | 200 |
| 2 | 1 | 2022 | 200 |
| 2 | 33 | 2023 | 200 |
| 2 | 36 | 2023 | 900 |
| 2 | 37 | 2023 | 400 |
| 2 | 40 | 2023 | 200 |
| 2 | 41 | 2022 | 200 |
| 3 | 8 | 2023 | 200 |
| 3 | 16 | 2021 | 200 |
| 3 | 33 | 2023 | 200 |
| 3 | 35 | 2021 | 200 |
| 3 | 36 | 2023 | 200 |
| 3 | 37 | 2023 | 1500 |
| 4 | 24 | 2021 | 200 |
| 4 | 36 | 2023 | 200 |
| 4 | 37 | 2023 | 800 |
| 4 | 38 | 2023 | 400 |
| 4 | 40 | 2021 | 200 |
| 4 | 51 | 2022 | 200 |
| 5 | 16 | 2021 | 200 |
| 5 | 33 | 2023 | 200 |
| 5 | 36 | 2022 | 200 |
| 5 | 36 | 2023 | 200 |
| 5 | 37 | 2023 | 400 |
| 5 | 46 | 2020 | 200 |
| 5 | 48 | 2020 | 200 |
| 6 | 6 | 2021 | 200 |
| 6 | 15 | 2023 | 200 |
| 6 | 24 | 2021 | 200 |
| 6 | 33 | 2023 | 200 |
| 6 | 36 | 2022 | 200 |
| 6 | 36 | 2023 | 400 |
| 6 | 37 | 2023 | 800 |
| 6 | 40 | 2023 | 200 |
| 6 | 48 | 2020 | 200 |
| 7 | 16 | 2021 | 200 |
| 7 | 20 | 2022 | 200 |

| | | | |
|---|---|---|---|
| 7 | 35 | 2023 | 200 |
| 7 | 36 | 2022 | 200 |
| 7 | 36 | 2023 | 400 |
| 7 | 37 | 2023 | 800 |
| 7 | 40 | 2021 | 200 |
| 7 | 48 | 2020 | 200 |
| 8 | 20 | 2022 | 200 |
| 8 | 36 | 2023 | 800 |
| 8 | 37 | 2023 | 400 |
| 8 | 40 | 2021 | 200 |
| 9 | 16 | 2021 | 200 |
| 9 | 20 | 2022 | 200 |
| 9 | 33 | 2023 | 200 |
| 9 | 36 | 2022 | 200 |
| 9 | 36 | 2023 | 600 |
| 9 | 37 | 2023 | 400 |
| 9 | 40 | 2023 | 200 |
| 9 | 46 | 2020 | 200 |
| 9 | 51 | 2022 | 200 |
| 10 | 6 | 2021 | 200 |
| 10 | 36 | 2023 | 600 |
| 10 | 37 | 2020 | 200 |
| 10 | 37 | 2023 | 600 |
| 10 | 41 | 2022 | 200 |
| 10 | 51 | 2022 | 200 |
| 11 | 11 | 2022 | 200 |
| 11 | 35 | 2021 | 200 |
| 11 | 36 | 2022 | 200 |
| 11 | 36 | 2023 | 1300 |
| 11 | 37 | 2020 | 200 |
| 11 | 37 | 2023 | 1600 |
| 11 | 38 | 2023 | 200 |
| 11 | 46 | 2020 | 200 |
| 11 | 48 | 2020 | 200 |
| 12 | 6 | 2021 | 200 |
| 12 | 8 | 2023 | 200 |
| 12 | 15 | 2023 | 200 |
| 12 | 20 | 2022 | 200 |
| 12 | 24 | 2021 | 200 |
| 12 | 35 | 2023 | 200 |
| 12 | 36 | 2022 | 200 |
| 12 | 36 | 2023 | 200 |
| 12 | 37 | 2023 | 800 |
| 12 | 40 | 2023 | 200 |

|  |  |  |  |
|---|---|---|---|
| 13 | 24 | 2023 | 200 |
| 13 | 36 | 2023 | 400 |
| 13 | 37 | 2023 | 600 |
| 13 | 38 | 2023 | 200 |
| 13 | 48 | 2020 | 200 |
| 14 | 6 | 2021 | 200 |
| 14 | 8 | 2023 | 200 |
| 14 | 11 | 2022 | 200 |
| 14 | 33 | 2023 | 200 |
| 14 | 35 | 2023 | 400 |
| 14 | 36 | 2023 | 1100 |
| 14 | 37 | 2020 | 200 |
| 14 | 37 | 2023 | 1300 |
| 14 | 38 | 2023 | 400 |
| 14 | 48 | 2020 | 200 |
| 14 | 51 | 2022 | 200 |
| 15 | 1 | 2022 | 200 |
| 15 | 15 | 2023 | 200 |
| 15 | 35 | 2021 | 200 |
| 15 | 35 | 2023 | 200 |
| 15 | 36 | 2022 | 200 |
| 15 | 36 | 2023 | 800 |
| 15 | 37 | 2023 | 1000 |
| 15 | 38 | 2023 | 200 |
| 15 | 40 | 2021 | 200 |
| 15 | 46 | 2020 | 200 |
| 15 | 48 | 2020 | 200 |
| 15 | 51 | 2022 | 200 |
| 16 | 11 | 2022 | 200 |
| 16 | 24 | 2023 | 200 |
| 16 | 37 | 2023 | 400 |
| 16 | 48 | 2020 | 200 |
| 17 | 1 | 2022 | 200 |
| 17 | 6 | 2021 | 200 |
| 17 | 11 | 2022 | 200 |
| 17 | 16 | 2021 | 200 |
| 17 | 24 | 2023 | 200 |
| 17 | 35 | 2023 | 500 |
| 17 | 36 | 2022 | 200 |
| 17 | 36 | 2023 | 1400 |
| 17 | 37 | 2023 | 1600 |
| 17 | 38 | 2023 | 200 |
| 17 | 51 | 2022 | 200 |
| 18 | 35 | 2023 | 200 |

| | | | |
|---|---|---|---|
| 18 | 36 | 2023 | 200 |
| 18 | 37 | 2023 | 1600 |
| 18 | 51 | 2022 | 200 |
| 19 | 33 | 2023 | 200 |
| 19 | 36 | 2023 | 200 |
| 19 | 37 | 2023 | 600 |
| 19 | 48 | 2020 | 200 |
| 20 | 1 | 2022 | 200 |
| 20 | 20 | 2022 | 200 |
| 20 | 35 | 2021 | 200 |
| 20 | 36 | 2023 | 1200 |
| 20 | 37 | 2023 | 1600 |
| 20 | 38 | 2023 | 200 |
| 20 | 40 | 2023 | 200 |
| 20 | 41 | 2022 | 200 |
| 21 | 35 | 2021 | 200 |
| 21 | 36 | 2023 | 200 |
| 21 | 37 | 2023 | 1000 |
| 21 | 38 | 2023 | 400 |
| 21 | 40 | 2021 | 200 |
| 21 | 40 | 2023 | 200 |
| 22 | 24 | 2023 | 200 |
| 22 | 33 | 2023 | 200 |
| 22 | 35 | 2023 | 200 |
| 22 | 36 | 2023 | 1200 |
| 22 | 37 | 2023 | 1300 |
| 22 | 46 | 2020 | 200 |
| 22 | 51 | 2022 | 200 |
| 23 | 11 | 2022 | 200 |
| 23 | 33 | 2023 | 200 |
| 23 | 35 | 2021 | 200 |
| 23 | 36 | 2023 | 600 |
| 23 | 37 | 2020 | 200 |
| 23 | 37 | 2023 | 600 |
| 23 | 41 | 2022 | 200 |
| 24 | 1 | 2022 | 200 |
| 24 | 8 | 2023 | 200 |
| 24 | 11 | 2022 | 200 |
| 24 | 20 | 2022 | 200 |
| 24 | 24 | 2021 | 200 |
| 24 | 35 | 2021 | 200 |
| 24 | 35 | 2023 | 200 |
| 24 | 36 | 2023 | 500 |
| 24 | 37 | 2023 | 800 |

| | | | |
|---|---|---|---|
| 24 | 38 | 2023 | 400 |
| 24 | 40 | 2021 | 200 |
| 24 | 41 | 2022 | 200 |
| 25 | 6 | 2021 | 200 |
| 25 | 11 | 2022 | 200 |
| 25 | 24 | 2021 | 200 |
| 25 | 35 | 2021 | 200 |
| 25 | 35 | 2023 | 200 |
| 25 | 36 | 2023 | 1500 |
| 25 | 37 | 2023 | 1600 |
| 25 | 40 | 2023 | 200 |
| 25 | 51 | 2022 | 200 |

## Output for question 6)

The view in question 6 should be called **PricePerDay** and be such that the output of

SELECT * FROM **PricePerDay** ORDER BY c_id,date;

when run on the Epature database (after inserting the test data given in an additional file) should be:

| c_id | date | cost |
|---|---|---|
| 1 | 2021-08-30 | 200 |
| 1 | 2022-05-22 | 200 |
| 1 | 2022-12-20 | 200 |
| 1 | 2023-09-03 | 200 |
| 1 | 2023-09-08 | 200 |
| 1 | 2023-09-11 | 500 |
| 1 | 2023-09-12 | 200 |
| 1 | 2023-09-13 | 200 |
| 1 | 2023-09-15 | 200 |
| 1 | 2023-09-17 | 200 |
| 1 | 2023-09-20 | 200 |
| 2 | 2022-01-09 | 200 |
| 2 | 2022-10-15 | 200 |
| 2 | 2023-08-20 | 200 |
| 2 | 2023-09-05 | 200 |
| 2 | 2023-09-08 | 200 |
| 2 | 2023-09-09 | 500 |
| 2 | 2023-09-11 | 200 |
| 2 | 2023-09-17 | 200 |
| 2 | 2023-10-05 | 200 |
| 3 | 2021-04-22 | 200 |
| 3 | 2021-08-30 | 200 |
| 3 | 2023-02-25 | 200 |
| 3 | 2023-08-20 | 200 |
| 3 | 2023-09-09 | 200 |
| 3 | 2023-09-11 | 400 |
| 3 | 2023-09-12 | 200 |
| 3 | 2023-09-13 | 200 |
| 3 | 2023-09-15 | 500 |
| 3 | 2023-09-17 | 200 |
| 4 | 2021-06-15 | 200 |
| 4 | 2021-10-10 | 200 |
| 4 | 2022-12-20 | 200 |
| 4 | 2023-09-09 | 200 |
| 4 | 2023-09-11 | 200 |
| 4 | 2023-09-12 | 200 |
| 4 | 2023-09-13 | 400 |

| | | |
|---|---|---|
| 4 | 2023-09-18 | 200 |
| 4 | 2023-09-20 | 200 |
| 5 | 2020-11-20 | 200 |
| 5 | 2020-12-05 | 200 |
| 5 | 2021-04-22 | 200 |
| 5 | 2022-09-05 | 200 |
| 5 | 2023-08-20 | 200 |
| 5 | 2023-09-09 | 200 |
| 5 | 2023-09-11 | 200 |
| 5 | 2023-09-12 | 200 |
| 6 | 2020-12-05 | 200 |
| 6 | 2021-02-10 | 200 |
| 6 | 2021-06-15 | 200 |
| 6 | 2022-09-05 | 200 |
| 6 | 2023-04-10 | 200 |
| 6 | 2023-08-20 | 200 |
| 6 | 2023-09-05 | 200 |
| 6 | 2023-09-09 | 200 |
| 6 | 2023-09-11 | 400 |
| 6 | 2023-09-15 | 400 |
| 6 | 2023-10-05 | 200 |
| 7 | 2020-12-05 | 200 |
| 7 | 2021-04-22 | 200 |
| 7 | 2021-10-10 | 200 |
| 7 | 2022-05-22 | 200 |
| 7 | 2022-09-05 | 200 |
| 7 | 2023-09-03 | 200 |
| 7 | 2023-09-05 | 200 |
| 7 | 2023-09-08 | 200 |
| 7 | 2023-09-12 | 200 |
| 7 | 2023-09-13 | 400 |
| 7 | 2023-09-15 | 200 |
| 8 | 2021-10-10 | 200 |
| 8 | 2022-05-22 | 200 |
| 8 | 2023-09-07 | 200 |
| 8 | 2023-09-08 | 400 |
| 8 | 2023-09-09 | 200 |
| 8 | 2023-09-13 | 200 |
| 8 | 2023-09-15 | 200 |
| 9 | 2020-11-20 | 200 |
| 9 | 2021-04-22 | 200 |
| 9 | 2022-05-22 | 200 |
| 9 | 2022-09-05 | 200 |
| 9 | 2022-12-20 | 200 |

| | | |
|---|---|---|
| 9 | 2023-08-20 | 200 |
| 9 | 2023-09-07 | 200 |
| 9 | 2023-09-08 | 200 |
| 9 | 2023-09-09 | 200 |
| 9 | 2023-09-12 | 200 |
| 9 | 2023-09-14 | 200 |
| 9 | 2023-10-05 | 200 |
| 10 | 2020-09-15 | 200 |
| 10 | 2021-02-10 | 200 |
| 10 | 2022-10-15 | 200 |
| 10 | 2022-12-20 | 200 |
| 10 | 2023-09-07 | 200 |
| 10 | 2023-09-09 | 400 |
| 10 | 2023-09-13 | 200 |
| 10 | 2023-09-15 | 200 |
| 10 | 2023-09-17 | 200 |
| 11 | 2020-09-15 | 200 |
| 11 | 2020-11-20 | 200 |
| 11 | 2020-12-05 | 200 |
| 11 | 2021-08-30 | 200 |
| 11 | 2022-03-18 | 200 |
| 11 | 2022-09-05 | 200 |
| 11 | 2023-09-04 | 200 |
| 11 | 2023-09-07 | 200 |
| 11 | 2023-09-08 | 500 |
| 11 | 2023-09-09 | 400 |
| 11 | 2023-09-11 | 400 |
| 11 | 2023-09-12 | 400 |
| 11 | 2023-09-13 | 200 |
| 11 | 2023-09-14 | 400 |
| 11 | 2023-09-17 | 200 |
| 11 | 2023-09-20 | 200 |
| 12 | 2021-02-10 | 200 |
| 12 | 2021-06-15 | 200 |
| 12 | 2022-05-22 | 200 |
| 12 | 2022-09-05 | 200 |
| 12 | 2023-02-25 | 200 |
| 12 | 2023-04-10 | 200 |
| 12 | 2023-09-03 | 200 |
| 12 | 2023-09-08 | 200 |
| 12 | 2023-09-11 | 200 |
| 12 | 2023-09-12 | 200 |
| 12 | 2023-09-13 | 200 |
| 12 | 2023-09-15 | 200 |

| | | |
|---|---|---|
| 12 | 2023-10-05 | 200 |
| 13 | 2020-12-05 | 200 |
| 13 | 2023-06-15 | 200 |
| 13 | 2023-09-05 | 200 |
| 13 | 2023-09-08 | 200 |
| 13 | 2023-09-11 | 200 |
| 13 | 2023-09-15 | 200 |
| 13 | 2023-09-17 | 200 |
| 13 | 2023-09-20 | 200 |
| 14 | 2020-09-15 | 200 |
| 14 | 2020-12-05 | 200 |
| 14 | 2021-02-10 | 200 |
| 14 | 2022-03-18 | 200 |
| 14 | 2022-12-20 | 200 |
| 14 | 2023-02-25 | 200 |
| 14 | 2023-08-20 | 200 |
| 14 | 2023-09-03 | 400 |
| 14 | 2023-09-05 | 200 |
| 14 | 2023-09-08 | 400 |
| 14 | 2023-09-09 | 500 |
| 14 | 2023-09-11 | 200 |
| 14 | 2023-09-12 | 500 |
| 14 | 2023-09-15 | 400 |
| 14 | 2023-09-16 | 200 |
| 14 | 2023-09-18 | 200 |
| 14 | 2023-09-20 | 200 |
| 15 | 2020-11-20 | 200 |
| 15 | 2020-12-05 | 200 |
| 15 | 2021-08-30 | 200 |
| 15 | 2021-10-10 | 200 |
| 15 | 2022-01-09 | 200 |
| 15 | 2022-09-05 | 200 |
| 15 | 2022-12-20 | 200 |
| 15 | 2023-04-10 | 200 |
| 15 | 2023-09-03 | 200 |
| 15 | 2023-09-05 | 400 |
| 15 | 2023-09-08 | 400 |
| 15 | 2023-09-12 | 200 |
| 15 | 2023-09-13 | 200 |
| 15 | 2023-09-14 | 200 |
| 15 | 2023-09-16 | 200 |
| 15 | 2023-09-17 | 200 |
| 15 | 2023-09-18 | 200 |
| 16 | 2020-12-05 | 200 |

| | | |
|---|---|---|
| 16 | 2022-03-18 | 200 |
| 16 | 2023-06-15 | 200 |
| 16 | 2023-09-13 | 200 |
| 16 | 2023-09-17 | 200 |
| 17 | 2021-02-10 | 200 |
| 17 | 2021-04-22 | 200 |
| 17 | 2022-01-09 | 200 |
| 17 | 2022-03-18 | 200 |
| 17 | 2022-09-05 | 200 |
| 17 | 2022-12-20 | 200 |
| 17 | 2023-06-15 | 200 |
| 17 | 2023-09-03 | 500 |
| 17 | 2023-09-04 | 200 |
| 17 | 2023-09-05 | 200 |
| 17 | 2023-09-08 | 500 |
| 17 | 2023-09-09 | 500 |
| 17 | 2023-09-11 | 400 |
| 17 | 2023-09-12 | 500 |
| 17 | 2023-09-13 | 200 |
| 17 | 2023-09-14 | 200 |
| 17 | 2023-09-15 | 200 |
| 17 | 2023-09-17 | 200 |
| 17 | 2023-09-20 | 200 |
| 18 | 2022-12-20 | 200 |
| 18 | 2023-09-03 | 200 |
| 18 | 2023-09-09 | 200 |
| 18 | 2023-09-11 | 400 |
| 18 | 2023-09-12 | 200 |
| 18 | 2023-09-13 | 200 |
| 18 | 2023-09-14 | 200 |
| 18 | 2023-09-15 | 400 |
| 18 | 2023-09-16 | 200 |
| 19 | 2020-12-05 | 200 |
| 19 | 2023-08-20 | 200 |
| 19 | 2023-09-07 | 200 |
| 19 | 2023-09-11 | 200 |
| 19 | 2023-09-15 | 400 |
| 20 | 2021-08-30 | 200 |
| 20 | 2022-01-09 | 200 |
| 20 | 2022-05-22 | 200 |
| 20 | 2022-10-15 | 200 |
| 20 | 2023-09-07 | 200 |
| 20 | 2023-09-08 | 500 |
| 20 | 2023-09-09 | 500 |

|  |  |  |
|---|---|---|
| 20 | 2023-09-11 | 500 |
| 20 | 2023-09-12 | 400 |
| 20 | 2023-09-13 | 200 |
| 20 | 2023-09-15 | 500 |
| 20 | 2023-09-16 | 200 |
| 20 | 2023-09-18 | 200 |
| 20 | 2023-10-05 | 200 |
| 21 | 2021-08-30 | 200 |
| 21 | 2021-10-10 | 200 |
| 21 | 2023-09-08 | 200 |
| 21 | 2023-09-11 | 200 |
| 21 | 2023-09-12 | 400 |
| 21 | 2023-09-15 | 200 |
| 21 | 2023-09-16 | 200 |
| 21 | 2023-09-18 | 200 |
| 21 | 2023-09-20 | 200 |
| 21 | 2023-10-05 | 200 |
| 22 | 2020-11-20 | 200 |
| 22 | 2022-12-20 | 200 |
| 22 | 2023-06-15 | 200 |
| 22 | 2023-08-20 | 200 |
| 22 | 2023-09-03 | 200 |
| 22 | 2023-09-05 | 200 |
| 22 | 2023-09-07 | 200 |
| 22 | 2023-09-08 | 400 |
| 22 | 2023-09-09 | 400 |
| 22 | 2023-09-11 | 200 |
| 22 | 2023-09-12 | 200 |
| 22 | 2023-09-13 | 400 |
| 22 | 2023-09-15 | 500 |
| 23 | 2020-09-15 | 200 |
| 23 | 2021-08-30 | 200 |
| 23 | 2022-03-18 | 200 |
| 23 | 2022-10-15 | 200 |
| 23 | 2023-08-20 | 200 |
| 23 | 2023-09-05 | 200 |
| 23 | 2023-09-08 | 200 |
| 23 | 2023-09-09 | 200 |
| 23 | 2023-09-13 | 200 |
| 23 | 2023-09-14 | 200 |
| 23 | 2023-09-15 | 200 |
| 24 | 2021-06-15 | 200 |
| 24 | 2021-08-30 | 200 |
| 24 | 2021-10-10 | 200 |

| | | |
|---|---|---|
| 24 | 2022-01-09 | 200 |
| 24 | 2022-03-18 | 200 |
| 24 | 2022-05-22 | 200 |
| 24 | 2022-10-15 | 200 |
| 24 | 2023-02-25 | 200 |
| 24 | 2023-09-03 | 200 |
| 24 | 2023-09-09 | 500 |
| 24 | 2023-09-11 | 400 |
| 24 | 2023-09-12 | 200 |
| 24 | 2023-09-14 | 200 |
| 24 | 2023-09-18 | 200 |
| 24 | 2023-09-20 | 200 |
| 25 | 2021-02-10 | 200 |
| 25 | 2021-06-15 | 200 |
| 25 | 2021-08-30 | 200 |
| 25 | 2022-03-18 | 200 |
| 25 | 2022-12-20 | 200 |
| 25 | 2023-09-03 | 200 |
| 25 | 2023-09-04 | 200 |
| 25 | 2023-09-05 | 200 |
| 25 | 2023-09-07 | 200 |
| 25 | 2023-09-08 | 500 |
| 25 | 2023-09-09 | 400 |
| 25 | 2023-09-11 | 500 |
| 25 | 2023-09-12 | 200 |
| 25 | 2023-09-13 | 200 |
| 25 | 2023-09-14 | 400 |
| 25 | 2023-09-15 | 400 |
| 25 | 2023-10-05 | 200 |