IS5102 Database Management Systems

Practical 2:
Database Management with SQLite

Due date: 13/11/2023 9:00PM

230013043

Task 1: Translation

Database schemas reflecting the given E-R-Model:

Schemas for Entity Sets:	
1.	staff(id, staff_name, email, street, city, postcode)
2.	phone(*id*, type, phone_number)
3.	station(station_name, town, *id*)
4.	service(service_number, *origin_station_name, destination_station_name*)
5.	stop(stop_name)
6.	service_time(*service_number* , start_time)
7.	manager(*id*, annual_salaray)
8.	driver(*id*, hourly_salary)

Schemas for Relationship Sets:	
9.	drives(*service_number*, *id*, hours_driven)
10.	on(service_number, stop_name, start_time, arrival_time, fare_from_origin)

Note that

- primary keys are underlined,
- foreign keys are written in *italics*.

Rationale:

1. Staff(id, staff_name, email, street, city, postcode)

Staff consists of several simple and some composite attributes. Address is a composite attribute. It is not listed separately, but instead we just list the components (Silberschatz, Korth, & Sudarshan, 2020, p. 265 f.). Additionally, staff consists of the multivalued attribute phone, for which we create a separate schema:

2. phone(*id*, type, phone_number)

For multivalued attributes, there is a separate table created which consists of the primary key of its originating relation (id from staff) as well as all its multivalued attributes (Silberschatz, Korth, & Sudarshan, 2020, p. 266). As a phone number is unique, I decided only to make number and not type a primary key. Every tuple in phone can be uniquely identified by the staff-ID and the phone-number (id, number), as described in the "time_slot"-example in the textbook (ibid, p. 267). The type of a phone number is not necessary to uniquely identify each tuple in this table, so it is not necessarily to include it to the primary key.

3. station(station_name, town, *id*)

Station consists of the primary key station_name, town, and id. We add id as foreign key to capture the "manages" relationship set, which links every station to a manager (Silberschatz, Korth, & Sudarshan, 2020, p. 270).

4. service(service_number, *origin_station_name, destination_station_name*)

> Service consists of service_number, which is its primary key. In addition to that, we add origin_station_name and destination_station_name as foreign keys, so that we catch the relationships destination and origin. The attributes origin_station_name and destination_station_name both reference station_name from the station-schema.

5. stop(stop_name)

> Stop consists only of one attribute, stop_name, which is its primary key.

6. service_time(*service_number* , start_time)

> service_time is a weak entity set. It consists of its discriminator, start_time, and the primary key of the identifying relationship set, service, which is service_number. Service_number is in italics because it is a foreign key.

7. manager(*id*, annual_salaray) and 8. driver(*id*, hourly_salary)

> Both manager and driver have id as their primary keys. Additionally, id is a foreign key because it references the staff relation.

9. drives(*service_number*, *id*, hours_driven)

> The primary key of the drives-schema consists of two attributes, service_number referencing the service-schema, and id, referencing the staff-schema. We add the descriptive attribute hours_driven.

10. on(*service_number*, *stop_name*, *start_time*, arrival_time, fare_from_origin)

> The primary key of "on" consists of service_number, stop_name, and start_time. We add the descriptive attributes arrival_time and far_from_origin to the schema. stop_name and start_time must be part of the primary key as they reference the two entity sets stop and service_time, which are connected by "on". However, we must also add service_number to the primary key to make every tuple in this relation uniquely identifyable. Imagine a scenario in which two different service lines have the same stop and departed at the same time. E.g., in larger cities like Glasgow or Edinburgh with many connections, it may happen that two bus lines depart from the same bus station at the same time. To differentiate between these two tuples, we need to include service_number.

The relationship sets destination, origin, and manages are all a many-to-one relationship with total participation. Therefore, we do not need to create a table for them.

The relationship set “of” is redundant as its attributes are start_time and service_number, which is already included in “on”. This is generally the case for a relationship set schema that connects a weak entity set with its corresponding strong entity set (Silberschatz, Korth, & Sudarshan, 2020, p. 270). Therefore, we do not need to create an additional table for “of”.

The order of the schemas listed above is important for the implementation in task 2. A foreign key can only reference an attribute from another table if the attribute was previously defined in this table. I found this out after my order was originally different and my SQL script didn't run.

Task 2: Data Definition Language

2.1 General Remarks

I used the Command Line Shell for SQLite to create a database for the schemas from task 1. At the top of the buses.sql-file, I set “PRAGMA foreign_keys = TRUE;”, as specified. In addition to that, I added .mode column so that the output of the commands is shown in columns (Command Line Shell For SQLite).

2.2 Creating Tables

After that, I started to create a table for every schema defined in task 1 and inserted values immediately after the table creation. As described before, the order in which tables are created is important when setting integrity constraints. Tables with a foreign key must be created after the table in which the referenced attribute is defined, otherwise the foreign key constraint of the referencing relation will fail.

Regarding data types, I decided to use VARCHAR for most of the attributes to allow a string with variable length of characters. For IDs, prices, and salaries, I decided to use NUMERIC to enforce a number with fixed length and specified number of decimal-digits.

In addition to that, I specified that all primary keys and foreign keys are NOT NULL to ensure referential integrity. In addition to that, I set the NOT NULL constraints for some other attributes which provide essential information, such as staff_name.

2.3 Inserting Values

For the staff-relation, I inserted ten managers (first digit of ID is “1”) and ten drivers (first digit of ID is “2”).

For the station-table, I inserted ten stations. As this is a many-to-one relationship with total participation of all tuples in station, I assigned each station to exactly one manager. In accordance with the many-to-one relationship, manager 1001 is in charge of two stations, while manager 1010 manages no station.

For the stop-table, I inserted all stations (i.e., origin and destination stations) and added all stops. Thus, every station is also a stop, but not every stop is a station. The stop-table therefore has more than ten rows.

For the service-table, I inserted 11 bus lines with their origin and destination station names.

The service_time-table includes the service number and the start times of every service. As this is a many-to-one relationship with total participation on both sides, every start time must be assigned to exactly one service number, and every service number can have one or more different service times. All my bus lines depart at least two times per day.

The drives-table is based on a many-to-many relationship. Therefore, a driver can be associated with several service numbers, and a service number can be associated with several drivers. As we have partial participation on both sides, it is possible that a driver does not drive any of the bus lines, and that a bus line has no driver.

The “on”-table shows for every bus line the stop, the start time of the service (i.e. departure time at origin station), its arrival time at the stop, and the fare from origin. Therefore, for every service line there are all stops for the different service times listed. As we have partial participation here, we not every stop in the stop-table must participate in the relationship. This is the case for the stop “Edinburgh Airport”, which is not listed in the “on”-table. However, I wondered about the partial participation on the service time side, because that would mean that not every service time needs to be linked to a stop. However, in my scenario, all start times are associated with a stop name.

Task 3: Data Manipulation Language

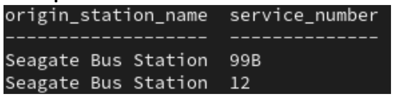
First, I tested all my tables defined and filled in task 2 by selecting all attributes from every table.

3.1 Queries:

1. List all services which have Seagate Bus Station in Dundee as their origin:

```
SELECT origin_station_name, service_number
FROM service
WHERE origin_station_name = "Seagate Bus Station";
```

Output:



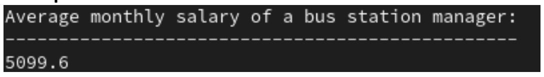
```
origin_station_name  service_number
-------------------  --------------
Seagate Bus Station  99B
Seagate Bus Station  12
```

Explanation: We retrieve the requested information from its table. I added origin_station_name so that we can clearly see that the bus lines start at Seagate Bus Station.

2. Calculate an average monthly salary of a bus station manager:

```
SELECT AVG (annual_salary/12) AS "Average monthly salary of a bus station
manager:"
FROM manager;
```

Output:



```
Average monthly salary of a bus station manager:
------------------------------------------------
5099.6
```

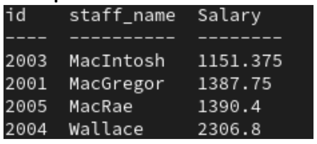
Explanation: We use aggregation to compute the average of annual salary divided by 12 (so we get the average monthly salary).

3. List the names of all drivers of services which have Edinburgh Bus Station in Edinburgh as their origin or destination, in increasing order of the amount to be paid to them for the hours driven:

```
SELECT
        staff.id,
        staff.staff_name,
        SUM ((driver.hourly_salary * drives.hours_driven)) AS "Salary"
FROM
        service
JOIN
        drives ON service.service_number = drives.service_number
JOIN
```

```
            staff ON drives.id = staff.id
    JOIN
            driver ON staff.id = driver.id
    WHERE
            (origin_station_name = "Edinburgh Bus Station")
            OR (destination_station_name = "Edinburgh Bus Station")
    GROUP BY
            staff.id
    ORDER BY
            Salary;
```

Output:



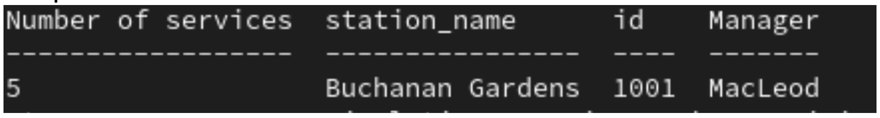
```
id    staff_name  Salary
----  ----------  --------
2003  MacIntosh   1151.375
2001  MacGregor   1387.75
2005  MacRae      1390.4
2004  Wallace     2306.8
```

Explanation: We need to join several tables together, because the information we need is stored in the tables service, drives, staff and driver. They are connected via service_number or id. Next, we filter the result by specifying that the origin or the destination is “Edinburgh Bus Station”. Finally, we group the result by staff_id to avoid duplicate rows in the output and we order the result by salary (ascending order is default, so we do not have to specify that).

4. List the manager of the most connected station, measured by the number of services which have
-- that station as their origin or destination.

```
    SELECT
            MAX (station_count) AS "Number of services",
            station.station_name, staff.id,
            staff.staff_name AS Manager
    FROM station
    JOIN (
            SELECT station.station_name, COUNT(*) AS station_count
            FROM station
            JOIN service ON (station.station_name = service.origin_station_name)
                    OR (station.station_name = service.destination_station_name)
            GROUP BY station.station_name
            ORDER BY station_count DESC
    ) AS connections ON station.station_name = connections.station_name
    JOIN staff ON station.id = staff.id;
```

Output:



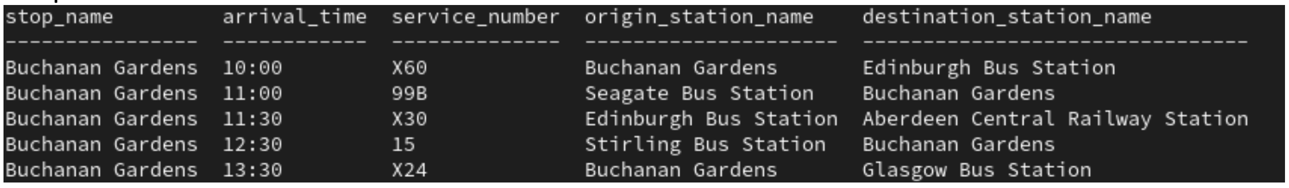
```
Number of services  station_name      id    Manager
------------------  ----------------  ----  -------
5                   Buchanan Gardens  1001  MacLeod
```

Explanation: My approach was first to create the subquery. The output of the subquery is a list of stations and their corresponding count of connections. The outer query will then find the maximum station_count and list this with staff_id and staff_name.

5. For the bus stop "Buchanan Gardens, St Andrews" list in the chronological order arrival times at this stop, origins, destinations, and service numbers of all bus services passing this stop between 10 am and 2 pm.

```
SELECT
        stop.stop_name,
        arrives.arrival_time,
        service.service_number,
        service.origin_station_name,
        service.destination_station_name
FROM
        stop
JOIN
        arrives ON stop.stop_name = arrives.stop_name
JOIN
        service ON arrives.service_number = service.service_number
WHERE
        stop.stop_name = "Buchanan Gardens"
        AND arrives.arrival_time >= "10:00"
        AND arrives.arrival_time <= "14:00"
ORDER BY
        arrives.arrival_time;
```

Output:



```
stop_name         arrival_time  service_number  origin_station_name    destination_station_name
----------------  ------------  --------------  ---------------------  ---------------------------------
Buchanan Gardens  10:00         X60             Buchanan Gardens       Edinburgh Bus Station
Buchanan Gardens  11:00         99B             Seagate Bus Station    Buchanan Gardens
Buchanan Gardens  11:30         X30             Edinburgh Bus Station  Aberdeen Central Railway Station
Buchanan Gardens  12:30         15              Stirling Bus Station   Buchanan Gardens
Buchanan Gardens  13:30         X24             Buchanan Gardens       Glasgow Bus Station
```

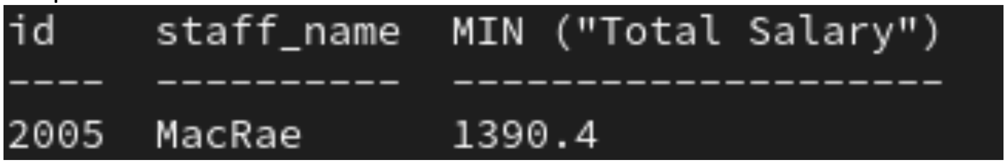
Explanation: We join all required tables together and specify in the WHERE clause that the stio-name is “Buchanan Gardens” and that the arrival time is larger than 10 am and smaller than 2 am. As I used 24-hours format, I used only “10:00” and “14:00”.

Additional own queries:

6. List ID and name of the driver with the minimum total salary.

```
SELECT driver.id, staff.staff_name, MIN ("Total Salary")
FROM driver
JOIN (
        SELECT
                driver.id,
                SUM (driver.hourly_salary * drives.hours_driven) AS "Total Salary"
        FROM driver
        JOIN drives ON driver.id = drives.id
        GROUP BY driver.id
) AS "Sum of salaries" ON driver.id = "Sum of salaries".id
JOIN staff ON driver.id = staff.id;
```

Output:



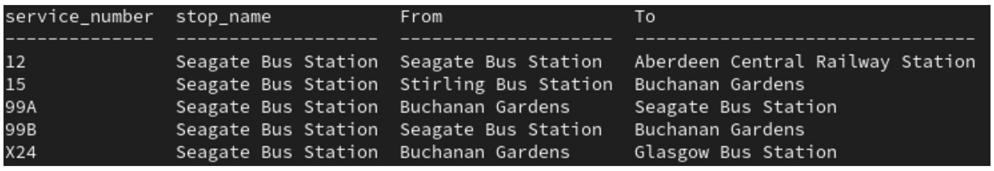
```
id    staff_name  MIN ("Total Salary")
----  ----------  --------------------
2005  MacRae      1390.4
```

Explanation: My approach was first to write and test the subquery, which returns a list of drivers with the sum of their salaries (“Total Salary”). After that, I created the outer query which finds the row with the minimum total salary in the subquery.

7. Which service lines pass the stop "Seagate Bus Station"? List service number as well as origin and destination.

```
SELECT
        service.service_number,
        arrives.stop_name,
        service.origin_station_name AS "From",
        service.destination_station_name AS "To"
FROM
        service
JOIN
        service_time ON service.service_number = service_time.service_number
JOIN
        arrives ON service_time.service_number = arrives.service_number
WHERE
        arrives.stop_name = "Seagate Bus Station"
GROUP BY
        service.service_number;
```

Output:



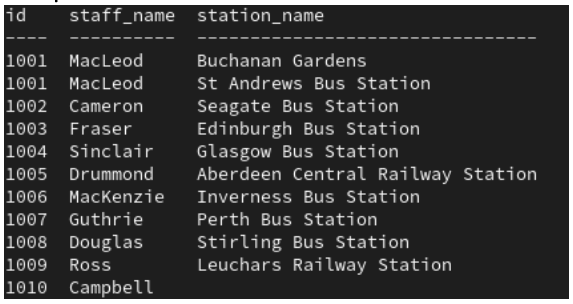
```
service_number  stop_name            From                  To
--------------  -------------------  --------------------  --------------------------------
12              Seagate Bus Station  Seagate Bus Station   Aberdeen Central Railway Station
15              Seagate Bus Station  Stirling Bus Station  Buchanan Gardens
99A             Seagate Bus Station  Buchanan Gardens      Seagate Bus Station
99B             Seagate Bus Station  Seagate Bus Station   Buchanan Gardens
X24             Seagate Bus Station  Buchanan Gardens      Glasgow Bus Station
```

Explanation: We retrieve the required information from three different tables (by joining them) and specify that stop_name must be "Seagate Bus Station", so that the output includes only rows with "Seagate Bus Station" as a stop. Finally, we group by service_number to avoid duplicate results.

8. List ID and name of all managers and the station name of the station they are managing, including managers who are not in charge of a station.

```
SELECT manager.id, staff.staff_name, station.station_name
FROM manager
JOIN staff ON manager.id = staff.id
LEFT JOIN station ON staff.id = station.id;
```

Output:



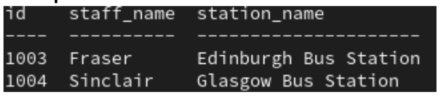
```
id    staff_name  station_name
----  ----------  --------------------------------
1001  MacLeod     Buchanan Gardens
1001  MacLeod     St Andrews Bus Station
1002  Cameron     Seagate Bus Station
1003  Fraser      Edinburgh Bus Station
1004  Sinclair    Glasgow Bus Station
1005  Drummond    Aberdeen Central Railway Station
1006  MacKenzie   Inverness Bus Station
1007  Guthrie     Perth Bus Station
1008  Douglas     Stirling Bus Station
1009  Ross        Leuchars Railway Station
1010  Campbell
```

Explanation: We use a LEFT (OUTER) JOIN for station because we want to include all managers, including those who have no station to manage. The OUTER JOIN allows us to include all rows in the manager relation and returns NULL as station name for every manager who is not in charge of a station.

9. Find the managers associated with the stations in Edinburgh and Glasgow.

```
SELECT staff.id, staff.staff_name, station.station_name
FROM staff
JOIN manager ON staff.id = manager.id
JOIN station ON manager.id = station.id
WHERE station.town = "Edinburgh" OR station.town = "Glasgow";
```

Output:



```
id    staff_name  station_name
----  ----------  ---------------------
1003  Fraser      Edinburgh Bus Station
1004  Sinclair    Glasgow Bus Station
```

Explanation: We have to join staff, manager and station and specify the town of the two stations.

3.2 Views:

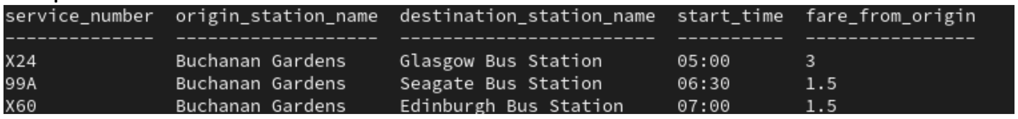
View 1: Customer

Customers should be able to see all tables with information about the service, but no sensitive data like names, contact details, address, and salary (data protection and privacy). However, joining all tables with the relevant information together would lead to such a large result, that it is difficult to find any meaning in this output. Therefore, I narrowed the following view down to a customer view that shows all lines that originate in Buchanan Gardens, at which time the service starts and how much the price is.

```
CREATE VIEW customer AS

        SELECT
                service.*,
                service_time.start_time,
                arrives.fare_from_origin
        FROM
                service
        LEFT JOIN
                service_time ON service.service_number = service_time.service_number
        LEFT JOIN
                arrives ON service_time.service_number = arrives.service_number
        GROUP BY service.service_number
        HAVING
                service.origin_station_name = "Buchanan Gardens"
        ORDER BY service_time.start_time
        ;

SELECT * FROM customer;
```

Output:


```
service_number  origin_station_name  destination_station_name  start_time  fare_from_origin
--------------  -------------------  ------------------------  ----------  ----------------
X24             Buchanan Gardens     Glasgow Bus Station       05:00       3
99A             Buchanan Gardens     Seagate Bus Station       06:30       1.5
X60             Buchanan Gardens     Edinburgh Bus Station     07:00       1.5
```

View 2: Finance department

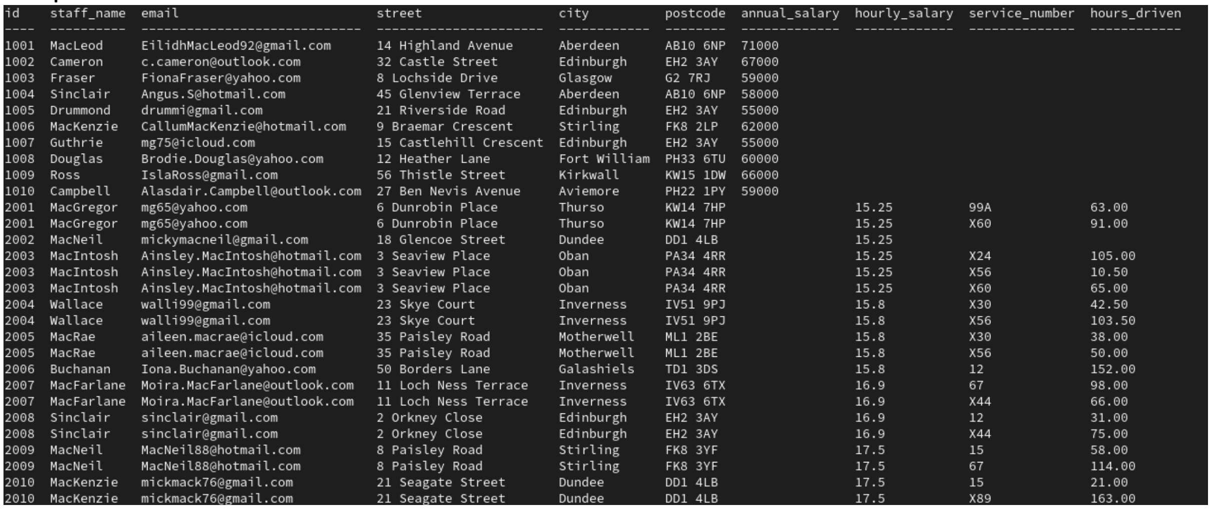
The finance department is responsible for payroll generation and therefore needs access to information in the staff, driver, drives, and manager relation:

```
CREATE VIEW finance AS

        SELECT
                staff.*,
                manager.annual_salary,
            driver.hourly_salary,
            drives.service_number,
            drives.hours_driven
      FROM staff
      LEFT JOIN manager ON staff.id = manager.id
      LEFT JOIN driver ON staff.id = driver.id
      LEFT JOIN drives ON driver.id = drives.id;

SELECT * FROM finance;
```

Output:



```
id    staff_name  email                          street                  city          postcode  annual_salary  hourly_salary  service_number  hours_driven
----  ----------  -----------------------------  ----------------------  ------------  --------  -------------  -------------  --------------  ------------
1001  MacLeod     EilidhMacLeod92@gmail.com      14 Highland Avenue      Aberdeen      AB10 6NP  71000
1002  Cameron     c.cameron@outlook.com          32 Castle Street        Edinburgh     EH2 3AY   67000
1003  Fraser      FionaFraser@yahoo.com          8 Lochside Drive        Glasgow       G2 7RJ    59000
1004  Sinclair    Angus.S@hotmail.com            45 Glenview Terrace     Aberdeen      AB10 6NP  58000
1005  Drummond    drummi@gmail.com               21 Riverside Road       Edinburgh     EH2 3AY   55000
1006  MacKenzie   CallumMacKenzie@hotmail.com    9 Braemar Crescent      Stirling      FK8 2LP   62000
1007  Guthrie     mg75@icloud.com                15 Castlehill Crescent  Edinburgh     EH2 3AY   55000
1008  Douglas     Brodie.Douglas@yahoo.com       12 Heather Lane         Fort William  PH33 6TU  60000
1009  Ross        IslaRoss@gmail.com             56 Thistle Street       Kirkwall      KW15 1DW  66000
1010  Campbell    Alasdair.Campbell@outlook.com  27 Ben Nevis Avenue     Aviemore      PH22 1PY  59000
2001  MacGregor   mg65@yahoo.com                 6 Dunrobin Place        Thurso        KW14 7HP                 15.25          99A             63.00
2001  MacGregor   mg65@yahoo.com                 6 Dunrobin Place        Thurso        KW14 7HP                 15.25          X60             91.00
2002  MacNeil     mickymacneil@gmail.com         18 Glencoe Street       Dundee        DD1 4LB                  15.25
2003  MacIntosh   Ainsley.MacIntosh@hotmail.com  3 Seaview Place         Oban          PA34 4RR                 15.25          X24             105.00
2003  MacIntosh   Ainsley.MacIntosh@hotmail.com  3 Seaview Place         Oban          PA34 4RR                 15.25          X56             10.50
2003  MacIntosh   Ainsley.MacIntosh@hotmail.com  3 Seaview Place         Oban          PA34 4RR                 15.25          X60             65.00
2004  Wallace     walli99@gmail.com              23 Skye Court           Inverness     IV51 9PJ                 15.8           X30             42.50
2004  Wallace     walli99@gmail.com              23 Skye Court           Inverness     IV51 9PJ                 15.8           X56             103.50
2005  MacRae      aileen.macrae@icloud.com       35 Paisley Road         Motherwell    ML1 2BE                  15.8           X30             38.00
2005  MacRae      aileen.macrae@icloud.com       35 Paisley Road         Motherwell    ML1 2BE                  15.8           X56             50.00
2006  Buchanan    Iona.Buchanan@yahoo.com        50 Borders Lane         Galashiels    TD1 3DS                  15.8           12              152.00
2007  MacFarlane  Moira.MacFarlane@outlook.com   11 Loch Ness Terrace    Inverness     IV63 6TX                 16.9           67              98.00
2007  MacFarlane  Moira.MacFarlane@outlook.com   11 Loch Ness Terrace    Inverness     IV63 6TX                 16.9           X44             66.00
2008  Sinclair    sinclair@gmail.com             2 Orkney Close          Edinburgh     EH2 3AY                  16.9           12              31.00
2008  Sinclair    sinclair@gmail.com             2 Orkney Close          Edinburgh     EH2 3AY                  16.9           X44             75.00
2009  MacNeil     MacNeil88@hotmail.com          8 Paisley Road          Stirling      FK8 3YF                  17.5           15              58.00
2009  MacNeil     MacNeil88@hotmail.com          8 Paisley Road          Stirling      FK8 3YF                  17.5           67              114.00
2010  MacKenzie   mickmack76@gmail.com           21 Seagate Street       Dundee        DD1 4LB                  17.5           15              21.00
2010  MacKenzie   mickmack76@gmail.com           21 Seagate Street       Dundee        DD1 4LB                  17.5           X89             163.00
```

Explanation: We can see that there are NULL values included. It is important to use LEFT JOIN instead of an INNER JOIN, because we have managers and drivers with different attributes. If we use JOIN instead of LEFT JOIN, the output would be empty/nothing, because we always have NULL values for this combination of attributes.

View 3: Driver

Every driver should have access to information about his/her salary, specific service lines driven and how many hours he/she was driving that line. This view should be customised for every individual driver. I have created a view for the driver with ID 2005:

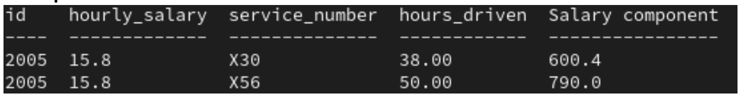
```
CREATE VIEW driver_2005 AS

      SELECT
            driver.*,
            drives.service_number,
            drives.hours_driven,
            SUM (driver.hourly_salary * drives.hours_driven) AS "Salary component"
      FROM driver
      JOIN drives ON driver.id = drives.id
      JOIN service ON drives.service_number = service.service_number
      WHERE driver.id = 2005
```

```
        GROUP BY drives.service_number;

SELECT * FROM driver_2005;
```

Output:



```
id    hourly_salary  service_number  hours_driven  Salary component
----  -------------  --------------  ------------  ----------------
2005  15.8           X30             38.00         600.4
2005  15.8           X56             50.00         790.0
```

Explanation: We see how many hours driver 2005 was driving for all lines he/she was driving. We see the salary component on the right side which is the product of hourly salary times hours driven.

Task 4: Reflection

This practical presented me with a number of challenges. The most difficult was creating the database in task 2, as I first had to familiarise myself with Linux and the terminal. In addition, small syntax errors were often the cause of errors and I spent a lot of time debugging. Overall, I would have found it more useful to have a finished database in order to have more time for the queries.

Some of the queries were very challenging for me. I realised that it makes sense to draw the connections between the individual tables visually in order to see which keys are used to connect them. In addition to that, I found it difficult to come up with my own queries. For query 7, I originally had a different question, but I realised that I could not implement it (see as a comment in the code).

To summarise, I think that I learned a lot from this practical, but the scope of the tasks was very large, so that in my opinion I had too little time to deal with the queries in detail.

Bibliography

Command Line Shell For SQLite. (2023, 11 1). Retrieved from https://sqlite.org/cli.html

Silberschatz, A., Korth, H., & Sudarshan, S. (2020). *Database System Concepts, 7th Edition.* New York: McGraw-Hill Education.