## SQL Coding (15 Points)

Pens Inc grows considerably in 3 years, and now maintains a database to store its orders. Their relational database contains the following four tables:

|  |  |
| --- | --- |
| users (  user\_id **INT PRIMARY KEY**  , first\_name **VARCHAR**(255)  , last\_name **VARCHAR**(255)  , date\_created **DATE**  , email **VARCHAR**(255)  ) | orders (  order\_id **INT PRIMARY KEY**  , user\_id **INT**  , date\_created **DATE**  , order\_value **FLOAT**  , city\_id **INT**  ) |
| city (  city\_id **INT PRIMARY KEY**  , city\_name **VARCHAR**(255)  , country\_id **INT**  ) | country (  country\_id **INT PRIMARY KEY**  , country\_name **VARCHAR**(255)  , currency\_code **VARCHAR**(255)  ) |

Please create an executable SQL statement to answer the questions below.

Note that you will be marked on whether the statement will run but also whether your statement is optimised and set out in a readable format.

Please indicate which SQL dialect you are using, e.g. “MySQL, PostgreSQL, Oracle…”

**MySQL**

1. *Count the number of customers, that placed at least 1 order, whose last\_name is “Scott”*  **[2 points]**

SELECT COUNT(DISTINCT(o.user\_id))

FROM deliveroo.orders AS o

LEFT JOIN deliveroo.users AS u

ON o.user\_id = u.user\_id

WHERE u.last\_name = 'Scott'

GROUP BY last\_name;

1. *Count the number of different cities where at least 1 order has been placed yesterday (do not hardcode the date)* ***[2 points]***

SELECT COUNT(DISTINCT(o.city\_id)) AS number\_cities

FROM deliveroo.orders AS o

LEFT JOIN deliveroo.city AS c

ON o.city\_id = c.city\_id

WHERE o.date\_created BETWEEN CURDATE() - INTERVAL 1 DAY AND CURDATE();

1. *Find the weekly order count for the city of Geneva for the last 8 weeks, and also the cumulative total.* ***[4 points]***

Desired output: *[week\_start, order\_count, cuml\_order\_count]*

SET @cuml\_sum := 0;

SELECT

DATE\_ADD(o.date\_created, INTERVAL -WEEKDAY(o.date\_created) DAY) week\_start,

COUNT(\*) AS order\_count,

(@cuml\_sum := @cuml\_sum + COUNT(\*)) AS cuml\_order\_count

FROM deliveroo.orders AS o

LEFT JOIN deliveroo.city AS c

ON o.city\_id = c.city\_id

WHERE c.city\_name = 'Geneva'

AND o.date\_created BETWEEN CURDATE() - INTERVAL 56 DAY AND CURDATE()

GROUP BY week\_start

ORDER BY week\_start ASC;

1. *Find the distribution of* ***first*** *order values per user in the UK, using a bin width of £10 and a maximum order value of £50. You can not assume that order ID field is ordered logically or sequentially.* ***[4 points]***

Desired output: *[order\_value\_bucket, order\_count]*

WITH FIRST\_ORDERS\_CTE AS (

SELECT

t1.user\_id,

t1.date\_created

FROM deliveroo.orders t1

INNER JOIN (

SELECT

user\_id,

MIN(date\_created) AS first\_date

FROM deliveroo.orders

GROUP BY user\_id

) t2 on t1.user\_id = t2.user\_id and t1.date\_created = t2.first\_date

)

SELECT

FLOOR(o.order\_value/10.00)\*10 as order\_value\_bucket,

COUNT(\*) AS order\_count

FROM deliveroo.users AS u

LEFT JOIN FIRST\_ORDERS\_CTE AS o

ON u.user\_id = o.user\_id

WHERE o.order\_value < 50

GROUP BY order\_value\_bucket ASC

ORDER BY order\_value\_bucket ASC

1. *Assuming that you have a query that runs very slowly, what are the steps that you would perform in order to identify the cause?* ***[3 points]***

The technique I would use in this instance with Microsoft SQL Server is to apply **Extended Events**.

Extended Events is a tracing feature built into the SQL Server database engine.

A GUI user interface was made available for Extended Event in the release of SQL Server 2012, even though the functionality dates back to 2005.

How to use Extended Events:

* To use Extended Events within SQL Server Management Studio, you go to the Object Explorer.
* After connecting to the database instance, you navigate to the Management folder, then on to Extended Events.
* Right click on the Sessions folder under 'Extended Events.' Here you can create a new extended event session by selecting 'New Session Wizard' from the context menu.
* Working through the wizard, you'll be prompted for a name, and whether you wish to have the session run on server startup. Fill these out and continue through the wizard.
* On the 'Choose Template' screen, select 'Do not use a template' and click the Next button.
* On the page to select which events you wish to capture, search for the word 'complete' and locate the rpc\_completed and sql\_batch\_completed events. Move them to the right side using the right arrow button.
* After selecting the events, the next page will let you view global fields to capture. The important one to capture here is the "sql\_text" field, as this is the actual text of the running query.
* On the session filters page of the wizard, you need to limit the number of commands captured. Do this in a similar way as you did in SQL Server profiler.
* In the field column, select sqlps.task\_execution\_time, set the operator to greator\_than\_unit64 and set the value column to 2500 which is 2.5 seconds.
* Select the storage location you wish to use, and finish the wizard.
* Once the wizard is closed, the session should automatically open in a new window. If it doesn't, right click on the session and select 'Watch Live Data' from the context menu. This will show you the long running queries on the system.
* Once you've identified the long-running queries, you can begin troubleshooting.