

## Data Warehousing Exercise 1

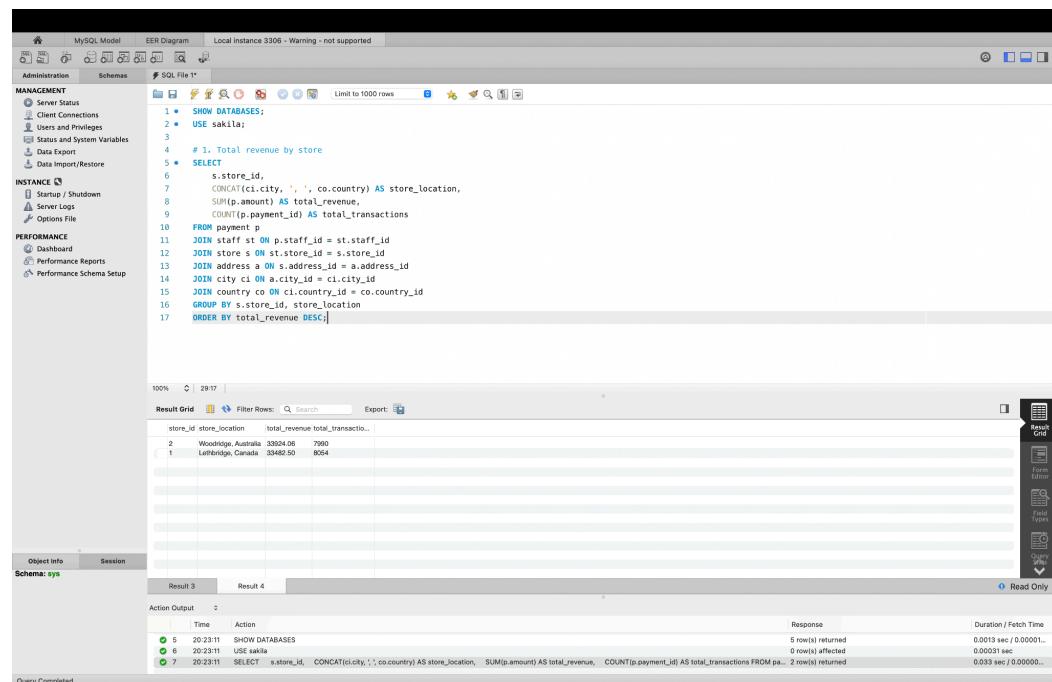
### 1. SQL Query Drill

Tool: MySQL Workbench

Dataset: I used the classic “Sakila” sample database:  
<https://dev.mysql.com/doc/sakila/en/sakila-introduction.html>

Tasks to complete:

- Total revenue by store:



The screenshot shows the MySQL Workbench interface. The left sidebar has sections for MANAGEMENT, INSTANCE, and PERFORMANCE. The main area contains a SQL editor with the following query:

```
1 • SHOW DATABASES;
2 • USE sakila;
3
4 # 1. Total revenue by store
5 • SELECT
6     s.store_id,
7     CONCAT(c1.city, ' ', co.country) AS store_location,
8     SUM(p.amount) AS total_revenue,
9     COUNT(p.payment_id) AS total_transactions
10    FROM payment p
11   JOIN staff st ON p.staff_id = st.staff_id
12   JOIN store s ON st.store_id = s.store_id
13   JOIN address a ON s.address_id = a.address_id
14   JOIN city c1 ON a.city_id = c1.city_id
15   JOIN country co ON c1.country_id = co.country_id
16 GROUP BY s.store_id, store_location
17 ORDER BY total_revenue DESC;
```

The results grid below the query shows two rows of data:

store_id	store_location	total_revenue	total_transactions
2	Woodridge, Australia	39294.06	7995
1	Lehnington, Canada	39482.50	8054

The bottom panel shows the query history:

Action	Time	Response	Duration / Fetch Time
SHOW DATABASES	20-23:11	6 row(s) returned	0.0013 sec / 0.00001...
USE sakila	20-23:11	0 row(s) affected	0.00031 sec
SELECT s.store_id, CONCAT(c1.city, ' ', co.country) AS store_location, SUM(p.amount) AS total_revenue, COUNT(p.payment_id) AS total_transactions FROM pa...	20-23:11	2 row(s) returned	0.033 sec / 0.00000...

- Average order value by customer segment

```

MySQL Model EER Diagram Local instance 3306 - Warning - not supported
Administration Schemas SQL File 1*
MANAGEMENT
INSTANCES
PERFORMANCE
SELECT
    # 2. Average order value (AOV) by customer segment
    # [SQL] segment customers into Bronze / Silver / Gold based on total spend
    WITH customer_spend AS (
        SELECT
            c.customer_id,
            CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
            SUM(p.amount) AS total_spent,
            COUNT(p.payment_id) AS transaction_count,
            CASE
                WHEN SUM(p.amount) >= 150 THEN 'Gold'
                WHEN SUM(p.amount) >= 100 THEN 'Silver'
                ELSE 'Bronze'
            END AS customer_segment
        FROM customer c
        JOIN payment p ON c.customer_id = p.customer_id
        GROUP BY c.customer_id, customer_name
    )
    SELECT
        customer_segment,
        COUNT(*) AS customers,
        ROUND(AVG(total_spent), 2) AS avg_total_spent,
        ROUND(AVG(total_spent / transaction_count), 2) AS avg_order_value
    FROM customer_spend
    GROUP BY customer_segment
    ORDER BY avg_order_value DESC;
    100% 2/343
Result Grid Filter Rows Search Export: 
customer_seg... customers avg_total_sp... avg_order_value
Gold       46   165.14  4.83
Silver     349  120.77  4.29
Bronze    204   86.58  3.95
Object Info Session Schema: sys Result 5 Result 6 Result 7 Read Only
Action Output Time Action Response Duration / Fetch Time
9 20-26-23 USE schema 0 row(s) affected 0.00027 sec
10 20-26-23 SELECT `store_id`, CONCAT(`city`,',',`country`) AS store_location, SUM(`amount`) AS total_revenue, COUNT(`payment_id`) AS total_transactions FROM `pa... 2 row(s) returned 0.031 sec / 0.00002...
11 20-26-23 WITH customer_spend AS ( SELECT c.customer_id, CONCAT(c.first_name,' ',c.last_name) AS customer_name, SUM(p.amount) AS total_spent, C... 3 row(s) returned 0.018 sec / 0.00003...
Query Completed

```

- Top 10 products by profit margin

```

MySQL Model EER Diagram Local instance 3306 - Warning - not supported
Administration Schemas SQL File 1*
MANAGEMENT
INSTANCES
PERFORMANCE
SELECT
    # 3. Top 10 films by profit margin
    # [SQL] Profit margin = (rental_rate - replacement_cost) is not meaningful - better: # profit generated vs replacement cost - Return on Inventory
    WITH film AS (
        SELECT
            f.title,
            f.rating,
            f.rental_rate,
            f.replacement_cost,
            COUNT(r.rental_id) AS times_rented,
            SUM(p.amount) AS total_revenue,
            ROUND((SUM(p.amount) - f.replacement_cost) / f.replacement_cost * 100, 2) AS ROI_percent
        FROM film f
        JOIN inventory i ON f.film_id = i.film_id
        JOIN rental r ON i.inventory_id = r.inventory_id
        JOIN payment p ON r.rental_id = p.rental_id
        GROUP BY f.film_id, f.title, f.rating, f.rental_rate, f.replacement_cost
        HAVING times_rented >= 10 -- only films rented at least 10 times
        ORDER BY ROI_percent DESC
        LIMIT 10
    )
    SELECT
        title,
        rating,
        rental_rate,
        replacement_c... times_rented,
        total_revenue,
        ROI_percent
    FROM film
    ORDER BY ROI_percent DESC;
    100% 2/103
Result Grid Filter Rows Search Export: 
title rating rental_rate replacement_c... times_rented total_revenue ROI_percent
TITAN'S JERK PG 4.99 11.99 29 201.71 1582.32
MAIDEN HOME PG 4.99 9.99 24 163.76 1539.24
VISITORS IN FRENCH NC-17 4.99 10.99 29 178.11 1501.11
STING PERSONAL NC-17 4.99 9.99 21 159.79 1469.50
WITCHES PANTS NC-17 4.99 10.99 30 173.70 1480.53
FELLOWS IN AUTUMN NC-17 4.99 9.99 26 148.70 1393.20
KISSING DOLLS R 4.99 9.99 20 147.80 1379.48
BOOGIE ANILIE R 4.99 11.99 29 163.70 1265.30
CLIFFHANGER R 4.99 12.99 28 172.11 1258.44
HEARTBREAKERS BRIGHT G 4.99 9.99 20 132.80 1209.33
Object Info Session Schema: sys Result 8 Result 9 Result 10 Result 11 Read Only
Action Output Time Action Response Duration / Fetch Time
14 20-27-27 SELECT `store_id`, CONCAT(`city`,',',`country`) AS store_location, SUM(`amount`) AS total_revenue, COUNT(`payment_id`) AS total_transactions FROM `pa... 2 row(s) returned 0.035 sec / 0.00006...
15 20-27-27 WITH customer_spend AS ( SELECT c.customer_id, CONCAT(c.first_name,' ',c.last_name) AS customer_name, SUM(p.amount) AS total_spent, C... 3 row(s) returned 0.035 sec / 0.00002...
16 20-27-27 SELECT `title`, `rating`, `rental_rate`, `replacement_cost`, COUNT(`rental_id`) AS times_rented, SUM(`amount`) AS total_revenue, ROUND((SUM(`amount`... 10 row(s) returned 0.040 sec / 0.0000...
Query Completed

```

- Bonus: Customers who spent > \$1,000 in 2024

MySQL Model EER Diagram Local instance 3306 - Warning - not supported

Administration Schemas SQL File 1\*

**MANAGEMENT**

- Server Status
- Client Connections
- Users and Privileges
- Starter and System Variables
- Data Events
- Data Import/Restore

**INSTANCE**

- Startup / Shutdown
- Server Logs
- Options File

**PERFORMANCE**

- Dashboard
- Performance Reports
- Performance Schema Setup

```

64  # 4. Bonus: Customers who spent > $1,000 in 2024
65  # (Note: Sakila data only goes up to 2006, so we'll use total lifetime spend > $200 as equivalent - very common tweak)
66  -- Real version for 2024 data (use this in real jobs)
67  -- SELECT
68  --
69  --   c.customer_id,
70  --   CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
71  --   c.email,
72  --   SUM(p.amount) AS total_spent_2024
73  --
74  -- FROM customer c
75  -- JOIN payment p ON c.customer_id = p.customer_id
76  -- WHERE YEAR(p.payment_date) = 2024
77  -- GROUP BY c.customer_id, customer_name, c.email
78  -- HAVING total_spent_2024 > 1000
79  -- ORDER BY total_spent_2024 DESC;
79
80  -- Sakila version (lifetime spend > $200 = top VIPs)
81  *
82  -- SELECT
83  --   c.customer_id,
84  --   CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
85  --   c.email,
86  --   SUM(p.amount) AS lifetime_spent,
87  --   COUNT(p.payment_id) AS transactions
87
88  FROM payment p
89  JOIN customer c ON c.customer_id = p.customer_id
90  GROUP BY c.customer_id, customer_name, c.email
91  HAVING lifetime_spent > 200
92  ORDER BY lifetime_spent DESC
92  LIMIT 20;
```

Result Grid Filter Rows: Search Export:

customer_id	customer_name	email	lifetime_spent	transactions
526	KARL SEAL	KARL_SEAL@sakilacustomer.org	231.59	45
148	ELEANOR HUNT	ELEANOR.HUNT@sakilacustomer.org	216.54	46

Object Info Session Schema: sys

Action Output 0 Time Action Response Duration / Fetch Time

Action	Time	Response	Duration / Fetch Time
20	20-29-58	WITH customer_spend AS ( SELECT c.customer_id, CONCAT(c.first_name,' ',c.last_name) AS customer_name, SUM(p.amount) AS total_spent, COUNT(p.payment_id) AS times_rented, SUM(p.amount) AS total_revenue, ROUND(SUM(p.amount)) AS total_spent_in_dollars ) SELECT * FROM customer_spend WHERE total_spent > 1000 ORDER BY total_spent DESC LIMIT 20;	0.027 sec / 0.000001...
21	20-29-58	SELECT @title, @rating, @rental_rate, @replacement_cost, COUNT(rental_id) AS times_rented, SUM(p.amount) AS total_revenue, ROUND(SUM(p.amount)) AS total_spent_in_dollars FROM customer_spend WHERE total_spent > 1000 ORDER BY total_spent DESC LIMIT 20;	0.040 sec / 0.000011...
22	20-29-58	SELECT c.customer_id, CONCAT(c.first_name,' ',c.last_name) AS customer_name, c.email, SUM(p.amount) AS lifetime_spent, COUNT(p.payment_id) AS transactions FROM payment p JOIN customer c ON c.customer_id = p.customer_id GROUP BY c.customer_id, customer_name, c.email HAVING lifetime_spent > 200 ORDER BY lifetime_spent DESC LIMIT 20;	0.014 sec / 0.000014...

Query Completed