

Data Warehousing Exercise 1

1. SQL Query Drill

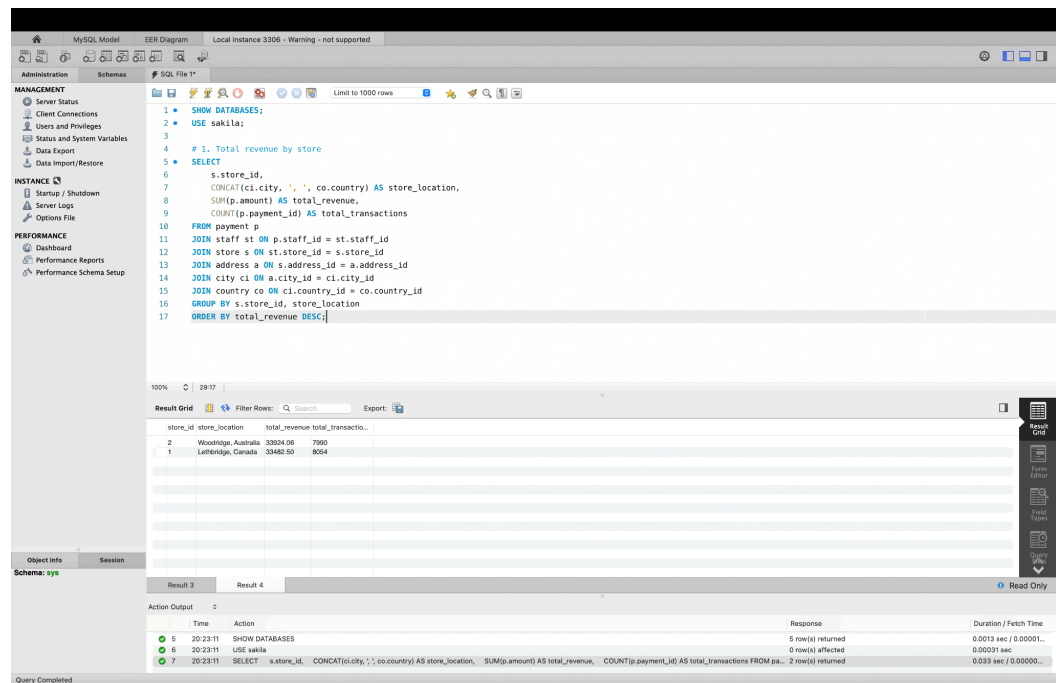
Tool: MySQL Workbench

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 SQL editor contains the following query:

```
1 • SHOW DATABASES;
2 • USE sakila;
3
4 # 1. Total revenue by store
5 • SELECT
6     s.store_id,
7     CONCAT(ci.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 ci ON a.city_id = ci.city_id
15 JOIN country co ON ci.country_id = co.country_id
16 GROUP BY s.store_id, store_location
17 ORDER BY total_revenue DESC;
```

The Results tab shows the following data:

store_id	store_location	total_revenue	total_transactions
2	Woodridge, Australia	33024.08	7980
1	Lethbridge, Canada	33482.50	8054

The bottom panel shows the Action Output log:

Action	Time	Response	Duration / Fetch Time
5	20:23:11	SHOW DATABASES	0 rows(s) returned
6	20:23:11	USE sakila	0 rows(s) affected
7	20:23:11	SELECT s.store_id, CONCAT(ci.city, ', ', co.country) AS store_location, SUM(p.amount) AS total_revenue, COUNT(p.payment_id) AS total_transactions FROM pa...	2 rows(s) returned

- Average order value by customer segment

The screenshot shows a MySQL Workbench interface with a SQL query and its results. The query is as follows:

```

17 ORDER BY total_revenue DESC;
18
19 # 2. Average order value (AOV) by customer segment
20 # We'll segment customers into Bronze / Silver / Gold based on total spend
21 WITH customer_spend AS (
22     SELECT
23         c.customer_id,
24         CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
25         SUM(p.amount) AS total_spent,
26         COUNT(p.payment_id) AS transaction_count,
27         CASE
28             WHEN SUM(p.amount) >= 150 THEN 'Gold'
29             WHEN SUM(p.amount) >= 100 THEN 'Silver'
30             ELSE 'Bronze'
31         END AS customer_segment
32     FROM customer c
33     JOIN payment p ON c.customer_id = p.customer_id
34     GROUP BY c.customer_id, customer_name
35 )
36 SELECT
37     customer_segment,
38     COUNT(*) AS customers,
39     ROUND(AVG(total_spent), 2) AS avg_total_spent,
40     ROUND(AVG(total_spent / transaction_count), 2) AS avg_order_value
41 FROM customer_spend
42 GROUP BY customer_segment
43 ORDER BY avg_order_value DESC;

```

The results are displayed in a table with the following columns: customer_segmen..., customers avg_total_sp..., avg_order_value. The data is as follows:

customer_segmen...	customers	avg_total_sp...	avg_order_value
Gold	48	185.14	4.83
Silver	349	120.77	4.29
Bronze	204	88.58	3.95

The Action Output section shows the following details:

Time	Action	Response	Duration / Fetch Time
20:26:23	USE sakila	0 row(s) affected	0.00027 sec
20:26:23	SELECT s.store_id, CONCAT(s.city, ' ', co.country) AS store_location, SUM(p.amount) AS total_revenue, COUNT(p.payment_id) AS total_transactions FROM pa...	2 row(s) returned	0.031 sec / 0.000002...
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 / 0.000003...

- Top 10 products by profit margin

The screenshot shows a MySQL Workbench interface with a SQL query and its results. The query is as follows:

```

40 ROUND(AVG(total_spent / transaction_count), 2) AS avg_order_value
41 FROM customer_spend
42 GROUP BY customer_segment
43 ORDER BY avg_order_value DESC;
44
45 # 3. Top 10 films by profit margin
46 # Profit margin = (rental_rate - replacement_cost) is not meaningful - better:
47 # revenue generated vs replacement cost - Return on Inventory
48 SELECT
49     f.title,
50     f.rating,
51     f.rental_rate,
52     f.replacement_cost,
53     COUNT(f.rental_id) AS times_rented,
54     SUM(p.amount) AS total_revenue,
55     ROUND((SUM(p.amount) - f.replacement_cost) / f.replacement_cost * 100, 2) AS ROI_percent
56 FROM film f
57 JOIN inventory i ON f.film_id = i.film_id
58 JOIN rental r ON i.inventory_id = r.inventory_id
59 JOIN payment p ON r.rental_id = p.rental_id
60 GROUP BY f.film_id, f.title, f.rating, f.rental_rate, f.replacement_cost
61 HAVING times_rented >= 10 -- only films rented at least 10 times
62 ORDER BY ROI_percent DESC
63 LIMIT 10;

```

The results are displayed in a table with the following columns: title, rating, rental_rate, replacement_c..., times_rent..., total_revenue, ROI_percent. The data is as follows:

title	rating	rental_rate	replacement_c...	times_rent...	total_revenue	ROI_percent
TRANS JERK	PG	4.99	11.99	29	201.71	1582.32
MADEN HOME	PG	4.99	9.99	24	183.76	1535.24
VIDEOTAPE ARSENC	NC-17	4.99	10.99	29	178.71	1508.11
STING PERSONAL	NC-17	4.99	9.99	21	159.79	1469.50
WITCHES PANIC	NC-17	4.99	10.99	30	173.70	1480.53
FELLOWSHIP AUTUMN	NC-17	4.99	9.99	26	149.74	1398.80
KISSING DOLLS	R	4.99	9.99	20	147.80	1371.48
BOOGIE AMELIE	R	4.99	11.99	29	163.70	1265.30
CLOSER HAND	R	4.99	12.99	28	172.72	1229.84
HEARTBREAKERS BRIGHT	G	4.99	9.99	20	132.80	1229.33

The Action Output section shows the following details:

Time	Action	Response	Duration / Fetch Time
20:27:47	SELECT s.store_id, CONCAT(s.city, ' ', co.country) AS store_location, SUM(p.amount) AS total_revenue, COUNT(p.payment_id) AS total_transactions FROM pa...	2 row(s) returned	0.032 sec / 0.000008...
20:27:47	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	0.028 sec / 0.000002...
20:27:47	SELECT f.film_id, f.rating, f.rental_rate, f.replacement_cost, COUNT(f.rental_id) AS times_rented, SUM(p.amount) AS total_revenue, ROUND(SUM(p.amou...	10 row(s) returned	0.040 sec / 0.000005...

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

The screenshot shows the MySQL Workbench interface. The top toolbar includes icons for file operations, database management, and execution. Below the toolbar are tabs for Administration, Schemas, and SQL File. The main window displays a SQL script with two queries. The first query calculates the total lifetime spend for customers who spent more than \$1,000 in 2024. The second query calculates the total lifetime spend for all customers, ordered by their lifetime spend in descending order. The bottom panel shows the 'Result Grid' with columns: customer_id, customer_name, email, lifetime_spend, and transactions. It contains three rows of data.

customer_id	customer_name	email	lifetime_spend	transactions
526	KARL BEAL	KARL_BEAL@sakila.com	221.55	45
148	ELEANOR HUNT	ELEANOR.HUNT@sakila.com	216.54	46