Sales Funnel Analysis – SQL Project

This document contains a complete SQL project consisting of 24 analytical queries performed on a sales funnel dataset. The queries cover data exploration, data quality checks, performance analysis, funnel validation, window functions, and preparation of a KPI-ready dataset for Power BI dashboarding.

1. Display all records from the raw sales funnel table.

```
SELECT * FROM sales_funnel;
```

2. Count the total number of activity records.

```
SELECT COUNT(*) AS total_count FROM sales_funnel;
```

3. Count the total number of unique IRs.

```
SELECT COUNT(DISTINCT(IR_ID)) AS Total_Unique_ids FROM sales_funnel;
```

4. List all Reporting Seniors and number of IRs working under each.

```
SELECT Reporting_Senior, COUNT(DISTINCT(name)) AS Total_Irs FROM sales_funnel GROUP BY
Reporting_Senior;
```

5. Identify records with missing values in critical columns.

```
SELECT * FROM sales_funnel WHERE IR_ID IS NULL OR activity_date IS NULL OR name IS NULL OR gender
IS NULL OR Reporting_Senior IS NULL OR INFO IS NULL OR Invites IS NULL OR POT IS NULL OR Closings
IS NULL;
```

6. Find IRs absent on certain days.

```
SELECT a.activity_date, i.IR_ID, i.name, i.Reporting_Senior FROM (SELECT DISTINCT IR_ID, name,
Reporting_Senior FROM sales_funnel) i CROSS JOIN (SELECT DISTINCT activity_date FROM
sales_funnel) a LEFT JOIN sales_funnel sf ON sf.activity_date = a.activity_date AND sf.IR_ID =
i.IR_ID ORDER BY a.activity_date, i.IR_ID;
```

7. Identify records where INFO is below the daily target of 5.

```
SELECT IR_ID, name, Reporting_Senior, COUNT(*) AS below_targets FROM sales_funnel WHERE INFO < 5
GROUP BY IR_ID, name, Reporting_Senior ORDER BY below_targets DESC;
```

8. Calculate percentage of records meeting INFO target.

```
SELECT (COUNT(CASE WHEN INFO >= 5 THEN 1 END) * 100 / COUNT(*)) AS percentage_meeting_target FROM
sales_funnel;
```

9. Detect funnel logic violations.

```
SELECT * FROM sales_funnel WHERE INFO < Invites OR Invites < POT OR POT < Closings;
```

10. Create a cleaned analytical dataset.

```
SELECT *, CASE WHEN INFO >= 5 THEN 'MATCH_TARGET' ELSE 'NOT_MATCHED_TARGET' END AS
info_target_status FROM sales_funnel WHERE IR_ID IS NOT NULL AND name IS NOT NULL AND
activity_date IS NOT NULL AND INFO IS NOT NULL AND INFO >= 0 AND Invites >= 0 AND POT >= 0 AND
Closings >= 0 AND INFO >= Invites AND Invites >= POT AND POT >= Closings;
```

11. Flag records as Valid, Below Target, or Invalid Funnel.

```
SELECT *, CASE WHEN INFO < Invites OR Invites < POT OR POT < Closings THEN 'Invalid Funnel' WHEN
INFO < 5 THEN 'Below Target' ELSE 'Valid' END AS record_status FROM sales_funnel;
```

12. Calculate daily totals for INFO, Invites, POT, and Closings.

```
SELECT activity_date, SUM(INFO) AS INFO, SUM(Invites) AS Invites, SUM(POT) AS POT, SUM(Closings)
AS Closings FROM sales_funnel GROUP BY activity_date;
```

13. Calculate overall funnel conversion rates.

```
SELECT ROUND(SUM(Invites)*100 / SUM(INFO), 2) AS info_to_invites_pct, ROUND(SUM(POT)*100 /
SUM(Invites), 2) AS invites_to_pot_pct, ROUND(SUM(Closings)*100 / SUM(POT), 2) AS
pot_to_closings_pct FROM sales_funnel;
```

14. Reporting Senior–wise total INFO and Closings.

```
SELECT Reporting_Senior, SUM(INFO) AS total_info, SUM(Closings) AS total_closings FROM
sales_funnel GROUP BY Reporting_Senior;
```

15. Reporting Senior–wise conversion rate.

```
SELECT Reporting_Senior, ROUND(SUM(Invites)*100 / SUM(INFO), 2) AS info_to_invites_pct,
ROUND(SUM(Closings)*100 / SUM(INFO), 2) AS info_to_closings_pct FROM sales_funnel GROUP BY
Reporting_Senior;
```

16. Rank IRs by total Closings.

```
SELECT IR_ID, name, SUM(Closings) AS total_closings, RANK() OVER (ORDER BY SUM(Closings) DESC) AS
rnk FROM sales_funnel GROUP BY IR_ID, name;
```

17. Top 3 IRs under each Reporting Senior.

```
WITH x AS (SELECT Reporting_Senior, IR_ID, name, SUM(Closings) AS total_closings, DENSE_RANK()
OVER (PARTITION BY Reporting_Senior ORDER BY SUM(Closings) DESC) AS rnk FROM sales_funnel GROUP
BY Reporting_Senior, IR_ID, name) SELECT * FROM x WHERE rnk <= 3;
```

18. IRs who consistently fail INFO target.

```
SELECT IR_ID, name FROM sales_funnel GROUP BY IR_ID, name HAVING MAX(INFO) < 5;
```

19. Gender-wise performance and conversion rates.

```
SELECT gender, SUM(INFO) AS total_info, SUM(Closings) AS total_closings, ROUND(SUM(Closings)*100
/ SUM(INFO), 2) AS conversion_pct FROM sales_funnel GROUP BY gender;
```

20. Days with unusually high or low Closings.

```
WITH daily AS (SELECT activity_date, SUM(Closings) AS closings FROM sales_funnel GROUP BY
activity_date) SELECT *, CASE WHEN closings > (SELECT AVG(closings) FROM daily) THEN 'High' WHEN
closings < (SELECT AVG(closings) FROM daily) THEN 'Low' ELSE 'Normal' END AS closing_level FROM
daily;
```

21. Cumulative Closings over time.

```
SELECT activity_date, SUM(Closings) AS daily_closings, SUM(SUM(Closings)) OVER (ORDER BY
activity_date) AS cumulative_closings FROM sales_funnel GROUP BY activity_date;
```

22. IRs performing above team average.

```
SELECT IR_ID, name, SUM(Closings) AS total_closings FROM sales_funnel GROUP BY IR_ID, name HAVING
SUM(Closings) > (SELECT AVG(total) FROM (SELECT SUM(Closings) AS total FROM sales_funnel GROUP BY
IR_ID) x);
```

23. Reporting Seniors with below-average team performance.

```
SELECT Reporting_Senior, SUM(Closings) AS total_closings FROM sales_funnel GROUP BY
Reporting_Senior HAVING SUM(Closings) < (SELECT AVG(total) FROM (SELECT SUM(Closings) AS total
FROM sales_funnel GROUP BY Reporting_Senior) x);
```

24. Final KPI-ready dataset for dashboarding.

```
CREATE OR REPLACE VIEW vw_sales_funnel_kpi AS SELECT activity_date, Reporting_Senior, IR_ID,
name, gender, INFO, Invites, POT, Closings, CASE WHEN INFO >= 5 THEN 1 ELSE 0 END AS target_met
FROM sales_funnel WHERE INFO >= Invites AND Invites >= POT AND POT >= Closings;
```