Project Blueprint

Business Problem

Customer support teams in service-oriented companies handle thousands of tickets daily through multiple channels — email, chat, phone, and social media.
Despite well-trained agents, businesses often face:

- **Long resolution times**
- **Inconsistent customer satisfaction (CSAT)**
- **Uneven agent performance**
- **Poor understanding of which issues or channels create the most friction**

These inefficiencies increase churn, operational cost, and reduce brand trust.

Problem Statement

“Reduce time-to-resolution and improve customer satisfaction by identifying high-impact issue categories, agent performance gaps, and ticket routing inefficiencies.”

Why This Works

The problem is **realistic and data-driven** — any company with customer support faces these KPIs.
It connects **business outcomes (CSAT, SLA)** with **operational levers (category, channel, agent)**.
It demonstrates both **technical skills (SQL, Power BI)** and **business acumen**.
It creates a story: “We analyzed 85,000 support tickets to pinpoint where efficiency and satisfaction drop, and designed a data-driven dashboard to optimize performance.”

Primary Business Questions

Theme	Key Questions	Impact
Efficiency	Which issue categories and channels have the longest resolution time?	Identify operational bottlenecks

Customer Satisfaction	What factors (category, channel, agent, shift, product) affect CSAT most?	Improve service quality
Agent Performance	Which agents or teams resolve tickets fastest or achieve highest CSAT?	Reward, train, or coach agents
Customer Impact	Are high-value customers facing lower satisfaction or delayed responses?	Prioritize premium customers
Process Optimization	Are certain shifts or supervisors showing patterns of poor performance?	Optimize routing and resource allocation

---

Project Blueprint

This project will be executed in 3 core phases (SQL + Power BI) and 1 optional Gen AI extension.
Each phase lists what you'll *do*, *learn*, and *how it ties back to the business problem.*

Phase 1 — Data Understanding & Cleaning (SQL – Foundation)

What We'll Do

- Import the raw CSV (≈85K records) into SQL Server as `raw_support_tickets`.
- Explore data types, missing values, and distinct counts.
- Parse and clean timestamps (`order_date_time`, `Issue_reported at`, `issue_responded`, `Survey_response_Date`).
- Handle missing values in `Customer_City`, `Item_price`, `connected_handling_time`.
- Standardize column names and create a clean, analysis-ready table: `stg_support_tickets`.

Techniques You'll Learn

- `CAST`, `TRY_CONVERT`, `COALESCE`, `ISNULL`
- Conditional replacements using `CASE`
- Creating staging tables and schemas
- `CHECK` and `DEFAULT` constraints for data quality

Business Value Tackled

Enables accurate KPI computation (resolution time, SLA breaches).
Improves data reliability for performance insights.

Phase 2 — Data Analysis & Insights (SQL – Analytics Layer)

What We'll Do

Use SQL to derive business insights by creating logical layers and reusable analytics.

A. Analytical KPIs & Views

- `vw_ticket_summary`: ticket volumes by category/channel/date
- `vw_agent_performance`: agent-wise avg resolution time & CSAT
- `vw_customer_impact`: trends for high-value customers (`Item_price > threshold`)
- `vw_channel_performance`: compare Email vs Chat vs Phone efficiency
- `vw_sla_compliance`: % tickets resolved under 48 hours

B. Advanced SQL Techniques

- **Window Functions** → rolling averages, rankings
- **CTEs** → simplify complex queries
- **Views** → reuse across Power BI
- **Stored Procedures** → generate monthly or category-specific reports
- **Joins & Subqueries** → link dimensions like `agent`, `supervisor`, `category`

Example Learnings

Concept	Example
Aggregation	`AVG(DATEDIFF(HOUR, issue_reported_at, issue_responded_at))`
Ranking	`RANK() OVER(ORDER BY avg_resolution_time)`
Trend Analysis	`DATEPART(WEEK, issue_reported_at)`
SLA	`CASE WHEN resolution_hours > 48 THEN 'Breach' ELSE 'Met' END`

Business Value Tackled

Identifies top pain areas (category/channel).
Reveals agent and shift performance gaps.
Provides a foundation for Power BI KPIs.

Phase 3 — Visualization & Reporting (Power BI)

What We'll Do

Connect Power BI to SQL views, model relationships, create DAX measures, and design an interactive dashboard.

Dashboard Plan & Suggested Visuals

Page	Visuals & KPIs	Business Problem Tackled
1. Executive Overview	Cards → Total Tickets, Avg Resolution Time, Avg CSAT, SLA Breach %Line Chart → Tickets & Avg CSAT over time	Overall performance trend
2. Category & Channel Insights	Clustered bar → Tickets by Category & ChannelMatrix → Category × Avg Resolution Time × CSAT	Root cause of inefficiencies

3. Agent & Shift Performance	Table → Agent leaderboard (Tickets, Avg Time, Avg CSAT)Bar → CSAT by Shift & TenureFilter → Supervisor/Manager	Workforce performance gaps
4. Customer Impact	Scatter → Item Price vs CSATMap → Ticket Volume by City	High-value customer experience
5. SLA & Operational Health	Line Chart → % SLA Met Over TimeGauge → SLA compliance	Process quality & efficiency
(Optional) 6. Remarks Insights	Text visuals, keyword clouds, or Gen AI summaries	Sentiment and root cause clues

DAX Concepts You'll Learn

- Calculated Columns: Resolution Hours, SLA Breach Flag
- Measures: Avg Resolution (Hrs), Avg CSAT, SLA Compliance %
- Time Intelligence: `TOTALYTD`, `DATESINPERIOD`

Phase 4 — Gen AI Extension (Later Stage)

Extension Idea 1: Ticket Summarization Bot

Goal: Automatically generate concise 1-line summaries of customer issues from the "Customer Remarks" column.

Approach:

- **Use an LLM (GPT-based model via Azure OpenAI or Python API)**
- **Generate summaries like:**
  "Login failed multiple times after update — request password reset assistance."
- **Store in SQL column `auto_summary` and visualize alongside ticket ID.**

Skills You'll Learn:

- **Prompt engineering for structured outputs**
- **Python–SQL integration**
- **Building an API pipeline for Power BI integration**

Business Value: Speeds up manager reviews and escalations.

Extension Idea 2: Smart Ticket Routing / Categorization

Goal: Use Gen AI to auto-assign a predicted category or urgency level from remarks text.

Example:
"Payment gateway timeout" **→ Predicted Category: "Billing Issue", Urgency: "High"**

Skills You'll Learn:

- **Text-to-category prompting**
- **Storing AI predictions in SQL**
- **Comparison analytics (AI category vs actual) in Power BI**

Business Value: Demonstrates end-to-end AI integration with analytics — a portfolio highlight.

GitHub Repository Structure

```
support-ticket-analytics/
│
├── data/
│   ├── raw/Customer_support_data.csv
│   └── cleaned/stg_support_tickets.csv
│
├── sql/
│   ├── 01_data_cleaning.sql
│   ├── 02_kpi_views.sql
│   ├── 03_stored_procedures.sql
│   └── 04_triggers_and_ctes.sql
│
├── powerbi/
│   ├── Support_Ticket_Dashboard.pbix
│   └── screenshots/
│
├── genai_extension/
│   ├── ticket_summarization_bot.ipynb
│   └── auto_categorization_prompt.ipynb
│
├── docs/
│   ├── business_problem.md
│   ├── data_dictionary.md
│   └── project_story.md
│
└── README.md
```

Learning Outcome Summary

Skill Area	You'll Gain
SQL Fundamentals	Cleaning, joins, aggregations, case logic
SQL Advanced	Window functions, CTEs, stored procedures, triggers
Power BI	Data modeling, DAX, KPI dashboards
Business Analytics	SLA, CSAT, agent performance optimization
Gen AI Concepts	Ticket summarization, category prediction
Portfolio Presentation	Professional repo, business storytelling

---

Technique - Problem Mapping

SQL Techniques → Business Problems Map

Project: Support Ticket Analytics Dashboard

1️⃣ Efficiency — *"Which issue categories and channels have the longest resolution time?"*

Component	How We'll Handle It	SQL Concepts Applied	Learning Value
Calculate Resolution Time	`DATEDIFF(SECOND, Issue_reported_at, issue_responded_at)/3600.0 AS resolution_hours`	**Derived Columns**, **CAST/CONVERT**	Compute key operational metrics
Find Avg Resolution by Category/Channel	Aggregate using `AVG()` and `GROUP BY category, channel_name`	**GROUP BY**, **Aggregation**	KPI building
Identify Longest Delay	`ORDER BY avg_resolution DESC`	**Sorting**, **Filtering**	Ranking by performance
Weekly Resolution Trend	Group by `DATEPART(WEEK, Issue_reported_at)`	**Date Functions**, **Time Series**	Trend analysis
Create reusable logic	Save as `vw_efficiency_trends`	**Views**	Modular query design

✅ **You'll Learn:**
How to calculate operational KPIs and turn raw timestamps into business insights.

✅ **Business Tie-In:**
Helps identify which issue types or channels (Chat, Email, Call) slow down resolution.

2️⃣ Customer Satisfaction — *"What factors affect CSAT most?"*

Component	How We'll Handle It	SQL Concepts Applied	Learning Value
Compute Avg CSAT by Category, Channel, Product	`AVG(CSAT_Score)`	**GROUP BY**, **Aggregate Functions**	Performance comparison
Compare High vs Low CSAT	`CASE WHEN CSAT_Score >=4 THEN 'High' ELSE 'Low' END`	**CASE WHEN**, **Conditional Logic**	Deriving satisfaction tiers
Correlate Resolution Time & CSAT	Use subquery or CTE to join resolution metrics with CSAT	**CTEs**, **Subqueries**	Multi-layer query logic
Identify Poor-Performing Segments	Filter CSAT < 3	**WHERE clause**	Analytical filtering
Reusable Summary	Create `vw_csat_drivers` view	**Views**	Reuse for Power BI connection

✅ ***You'll Learn:***
Connecting customer sentiment with operational data and how to layer calculations.

✅ ***Business Tie-In:***
Pinpoints which combinations of product/channel/issue degrade satisfaction.

3️⃣ *Agent Performance — "Which agents resolve faster and deliver higher CSAT?"*

Component	How We'll Handle It	SQL Concepts Applied	Learning Value
Agent Summary	Compute avg resolution & CSAT by agent	**GROUP BY agent_name**	Aggregation by dimension
Leaderboard	`RANK() OVER(ORDER BY avg_resolution ASC)`	**Window Functions (RANK, ROW_NUMBER)**	Advanced analytics
Rolling 7-Day Avg	`AVG(...) OVER (ORDER BY date ROWS 6 PRECEDING)`	**Window Functions (Rolling)**	Trend smoothing

Agent Category Mix	*Join agent table with ticket categories*	***JOINs*, *LEFT JOIN***	*Combining datasets*
Save results	`CREATE VIEW vw_agent_performance`	***Views***	*Reuse for dashboards*
Monthly refresh	*Wrap logic in stored proc* `sp_refresh_agent_metrics`	***Stored Procedures***	*Automation & modularity*

✅ ***You'll Learn:***
Ranking, window functions, stored procedures — the backbone of real analytics pipelines.

✅ ***Business Tie-In:***
Helps management identify top/bottom performers and patterns by shift or experience.

4️⃣ *Customer Impact — "Are high-value customers (Item_price) getting worse service?"*

Component	***How We'll Handle It***	***SQL Concepts Applied***	***Learning Value***
Create High-Value Flag	`CASE WHEN Item_price > 500 THEN 'High Value' ELSE 'Regular' END`	***CASE WHEN*, *Derived Columns***	*Business-driven data creation*
Join CSAT + Resolution	*Combine metrics from previous CTEs*	***CTEs*, *JOINs***	*Data model thinking*
Avg Resolution by Value Segment	`AVG(resolution_hours)` *grouped by high/regular*	***GROUP BY*, *Aggregate***	*Comparing customer groups*
SLA Comparison	`CASE WHEN resolution_hours > 48 THEN 'Breach' ELSE 'Met' END`	***Conditional Logic*, *KPI Flag***	*Operational KPIs*
View creation	`vw_customer_impact`	***Views***	*Reusability for BI layer*

✅ ***You'll Learn:***
How to derive business categories (segmentation) and test hypotheses.

✅ ***Business Tie-In:***
Reveals whether premium customers experience slower service — crucial for retention.

5️⃣ *Process Optimization — "Which shifts, supervisors, or managers show patterns of poor performance?"*

Component	***How We'll Handle It***	***SQL Concepts Applied***	***Learning Value***
Combine Hierarchy	*Aggregate by* `Manager`, `Supervisor`, `Agent_Shift`	***GROUP BY***, ***Joins***	*Hierarchical rollups*
Trend by Shift	*Compare avg CSAT & resolution*	***CASE WHEN***, ***Pivot Logic***	*Comparative metrics*
Identify Top 10 Problem Teams	*Use* `RANK()` *over grouped averages*	***Window Functions***, ***CTEs***	*Analytical ranking*
Build Monthly Refresh	*Store as* `sp_shift_performance_monthly`	***Stored Procedure***, ***Dynamic SQL***	*Automation practice*
Audit New Tickets	*Create trigger: if new ticket inserted → log insert in* `ticket_audit_log`	***Triggers***	*System monitoring*

✅ ***You'll Learn:***
Complex aggregation, procedural SQL, and triggers for automation.

✅ ***Business Tie-In:***
Reveals structural inefficiencies (specific teams, shifts, or leadership chains).